I. Văduva V. Florescu
D. Farcaș Gh. Popa N. Mihoc

Autoinstruire în programare

- Date, algoritmi
- Limbajul COBOL
- Depanare, cartele de comandă
- Programe complexe

editura tehnică

CUPRINS

RECOMANDARE IMPORTANTĂ PENTRU CITITORII CĂRȚII.

Citiți întrebările și completați (cu creionul) răspunsurile Dvs., păstrînd răspunsurile autorilor acoperite (cu o mică foaie de hîrtie). Apoi, comparînd soluțiile date, puteți să verificați imediat și permanent stadiul însușirii cunoștințelor expuse!

LECŢIA 1 STRUCTURA, FUNCŢIONAREA ŞI PROGRAMAREA CALCULATOARELOR ELECTRONICE

- configuraţii de calcul
- suporturi tehnice de date
- limbaje de programare
- compilare
- sistem de operare
- programare: noţiune şi etape

1.1 Aveţi noţiuni generale despre componentele unui calculator electronic?
DA → 1.21.
NU → citiţi paragrafele 1.2—1.20*

1.2 Un *calculator electronic* are ca principale componente (fig. 1.1):

— *memoria centrală* (principală) în care sînt memorate programele şi datele care fac obiectul prelucrării;

— *unitatea centrală*, care analizează şi execută programele** formate din secvenţe de instrucţiuni; în componenţa ei intră:

• *unitatea de comandă* care analizează, controlează şi comandă executarea instrucţiunilor din program;

• *unitatea aritmetică şi logică* care execută operaţiile aritmetice şi logice;

— *sistemul de intrări/ieşiri* care controlează şi execută operaţiile de intrare/ieşire ce au ca obiectiv transferul datelor de pe suporturi tehnice în memorie şi respectiv înregistrarea datelor pe suporturi tehnice de date***.

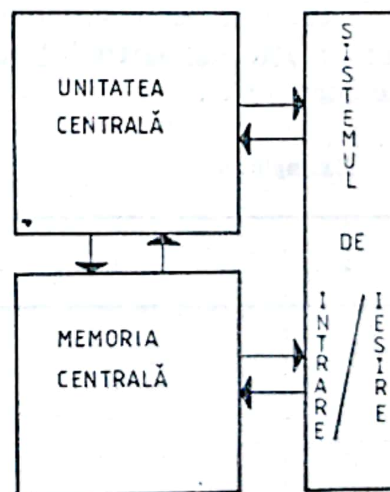


Fig. 1.1.

1.3 *Memoria centrală* este construită din reţele de inele de ferită aşezate în mai multe plane. Fiecare inel de ferită poate avea două stări şi, prin urmare, reprezenta una din valorile binare „0" sau „1". Un inel de ferită permite memorarea unei poziţii binare, sau altfel spus a unui bit. Capacitatea memoriei centrale se exprimă în kiloocteţi (K) unde:
1 K = 1 024 octeţi iar 1 octet = 8 biţi.

---

* Pentru aprofundare consultaţi lucrarea V. Popescu, „Instruire programată în calculatoare numerice", din aceeaşi serie.

** Definim intuitiv programul ca fiind o listă de operaţii pe care calculatorul le poate executa. Un element din listă se cheamă instrucţiune.

*** În lucrare vom spune şi prescurtat suport tehnic

Memoria centrală a calculatorului FELIX C-256 poate avea de la 1 la 4 blocuri; fiecare bloc avînd de regulă 64 K. Un bloc de memorie complet este alcătuit din patru module de capacitate 16 K fiecare.

Informaţiile sînt memorate sub forma succesiunilor de poziţii binare numite şi *locaţii*.

Poziţia locaţiei în cadrul memoriei dă *adresa locaţiei*. Operaţiile de intrare/ieşire au în vedere conţinutul locaţiei.

Octetul constituie o locaţie şi în acelaşi timp cea mai mică unitate adresabilă (fiecare octet se localizează în memoria centrală printr-o adresă).

1.4 Calculatorul execută programe înregistrate în memoria centrală. Acestea au efect asupra datelor care de asemenea trebuie, în momentul în care sînt supuse prelucrării, să se găsească în memoria centrală. Rezultatele sînt aşezate şi ele în memoria centrală, pe măsura obţinerii lor.

1.5 În memoria calculatorului datele au o reprezentare binară.

Calculatoarele din gama FELIX C admit mai multe forme de reprezentare a datelor, dintre care enumerăm pe cele utilizate în condiţiile folosirii limbajului COBOL:

— şir de caractere;
— zecimal — despachetat;
— zecimal — împachetat;
— în virgulă mobilă;
— binară (în cod complementar).

1.6 *Forma şir de caractere* se utilizează pentru reprezentarea caracterelor alfanumerice admise în prelucrare conform sistemului de codificare EBCDIC (vezi anexa 1).

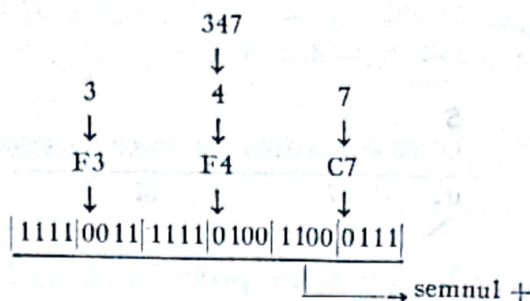
Fiecărui caracter EBCDIC i se asociază mai întîi o valoare hexazecimală formată din două simboluri (numere), iar fiecărui caracter hexazecimal o combinaţie de patru biţi conform corespondenţei între cele două sisteme de numeraţie.

Exemplu:

Caractere EBCDIC	F	E	L	I	X	C	–	2	5	6
Valoarea hexazecimală	C6	C5	D3	C9	E7	C3	6D	F2	F5	F6
Valoarea binară	11000110	11000101	11010011	11001001	11100111	01100011	01101101	11110010	11110101	11110110

1.7 *În forma de reprezentare zecimal — despachetată* fiecare cifră zecimală în cadrul unui număr este reprezentată printr-un grup de 8 cifre binare; primele 4 avînd valoarea 1111 (în hexazecimal F) cu excepţia ultimei cifre pentru care primele 4 cifre binare reprezintă valoarea semnului algebric.

Exemplu:



1.8 *În forma de reprezentare zecimal—împachetată* cifrelor zecimale și semnului algebric (+ sau —) le corespund grupe de 4 cifre binare:

Exemplu: 1981

1	9	8	1	C
0001	1001	1000	0001	1100

→ semnul +

sau în formă pe octeți:

|0000|0001|1001|1000|0001|1100|

unde: C este un caracter pentru a preciza semnul + ; D și B sînt pentru semnul —.

Avantajul acestei forme de reprezentare este dat de economia de memorie ce se poate realiza pentru date numerice prezentate în volum mare și de lungime (ordin) mare.

Lungimea zonei de memorie rezervată pentru un număr se stabilește după formula:

$$L = \left[\frac{l}{2}\right] + 1$$

unde l reprezintă lungimea datei.

Așa cum o să vedem din conținutul acestei lucrări, realizarea operațiilor de calcul implică conversia operanzilor la aceeași formă de reprezentare, corespunzătoare operatorului aritmetic al unității de comandă și prelucrare.

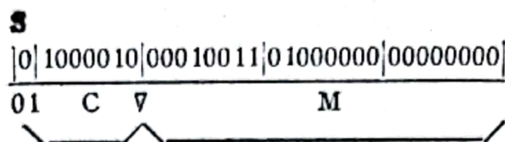
1.9 *Forma de reprezentare în virgulă mobilă* este utilizată pentru a putea supune prelucrării date de lungime foarte mare sau foarte mică și este caracteristică valorilor ce reprezintă numere reale (a căror lungime depășește 18 caractere).

În această formă de reprezentare prima cifră binară desemnează semnul (0 pentru valori pozitive și 1 pentru valori negative), următoarele 7 cifre binare caracteristica iar ultimele 24 (sau 56 cifre binare în cazul virgulei mobile cu dublă precizie) mantisa. Prin urmare numerele în virgulă mobilă sînt reprezentate pe 4 sau 8 locații (octeți). Numărul este exprimat în baza 16 (în heazecimal).

Exemplu de reprezentare în virgulă mobilă (în simplă precizie):

$N = 19{,}25_{10} = 13{,}4_{16} = 0{,}134 \times 16^2$

$M = 0{,}134_{16}$; $C = 64 + 2 = 66_{10} = 42_{16}$



La calcularea valorilor lui C se procedează astfel:

dacă $M < 0$ $\quad C = 63 -$ exponent

dacă $M > 0$ $\quad C = 64 +$ exponent

Mantisa se reprezintă în cod complementar, iar caracteristica (care este mereu pozitivă) în cod direct.

Un număr în dublă precizie se deosebește de cel în simplă precizie prin lungimea mantisei.

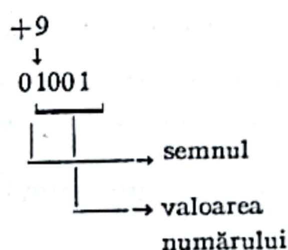
1.10 *Reprezentarea în binar* (sau cod complementar) se folosește pentru numere întregi pozitive sau negative.

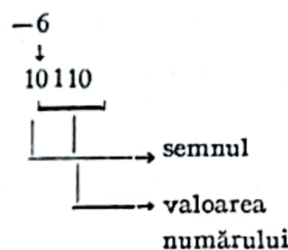
Valoarea binară asociată unui număr în bază 10 se structurează astfel:

— prima cifră binară indică semnul;

— următoarele *n* cifre binare indică numărul.

Exemple:





Această formă permite economisirea atît a spațiului de memorie cît și a timpului de prelucrare deoarece, așa cum o să vedem în lecțiile următoare, calculatorul efectuează calculele aritmetice într-un timp mai scurt dacă operanzii sînt memorați în binar.

Spre exemplu pentru un număr care conține între 5 și 9 caractere sînt suficienți 4 octeți.

1.11 În literatura de specialitate datele memorate în 2, 4 sau 8 octeți mai sînt numite și semicuvinte, cuvinte și respectiv dublu-cuvinte.

Așa cum o să vedem în lecțiile care urmează, prin extensie, se poate vorbi despre un mod de organizare a memoriei pe caractere (în cazul datelor reprezentate în formă șir de caractere, zecimal-despachetat și zecimal-împachetat), sau pe *semicuvinte, cuvinte* și *dublu cuvinte,* (pentru datele reprezentate în formă binară, virgulă flotantă în simplă precizie și virgulă flotantă în dublă precizie). O suită de locații (octeți) succesive de memorie desemnează o *zonă de memorie.* În procesul de prelucrare automată a datelor vom distinge *numele zonei de memorie* (utilizat pentru referențierea datelor) de *conținutul acesteia.*

Într-o instrucţiune de prelucrare se precizează numele zonei de memorie iar prelucrarea, operează asupra conţinutului acesteia.

1.12 *Î.* Principalele forme de reprezentare a datelor în memoria calculatorului sînt :

. . . .
. . . .
. . . .

R. şir de caractere
zecimal-despachetat
zecimal-împachetat
virgulă mobilă (în simplă şi dublă precizie)
binară (cod complementar)

1.13 *Î.* În forma de reprezentare zecimal-despachetată fiecare cifră zecimală se memorează a) . . . . în timp ce în reprezentarea zecimal-împachetată b) . . . .
. . . .
. . . .

R. a) într-un octet

b) cîte două cifre zecimale se memorează într-un octet, cu excepţia ultimei cifre din cadrul numărului care se memorează împreună cu semnul algebric pe ultimul octet.

1.14 Volumul mare de date şi programe implicate în operaţiile de prelucrare automată impun folosirea şi a memoriei auxiliare care se compune din discuri magnetice, benzi magnetice etc. (fig. 1.2).

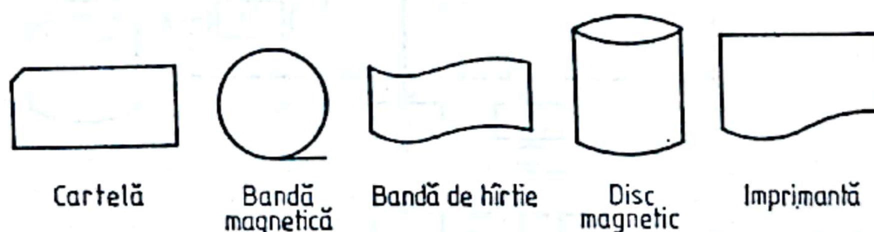


Fig. 1.2.

1.15 Pentru a comunica cu mediul exterior calculatoarele electronice dispun de *unităţi periferice* (numite în lucrare şi echipamente periferice). Între unitatea centrală şi unităţile periferice se interpun *unităţile de schimburi* şi *unităţile de legături* specializate în gestionarea transferului de date. Prin urmare calculatorul este constituit dintr-un ansamblu de unităţi distincte. În limbajul uzual ansamblul echipamentelor cuplate la unitatea centrală este desemnat prin termenul *configuraţie* de calcul (fig. 1.3.) *

* Linia întreruptă delimitează componentele principale de periferice.

Funcţionarea unităţilor din configuraţie * este dictată prin programul conţinut în memoria centrală.

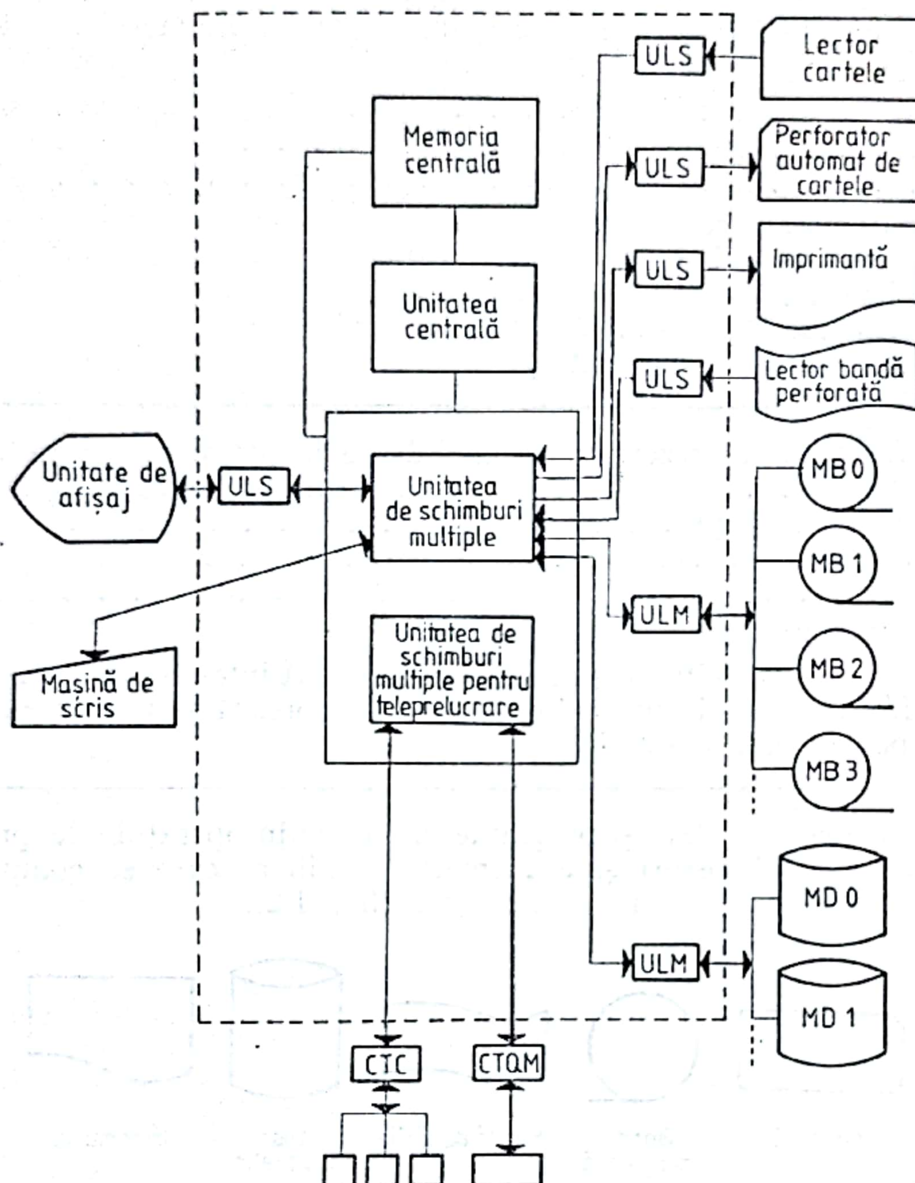


Fig. 1.3.

Unitatea de comandă, analizează instrucţiunile din program şi lansează ordinile de execuţie către *unitatea aritmetică şi logică* dacă este vorba de prelucrare sau către *unitatea de schimburi* dacă este vorba de transfer de date în/din exterior.

* ULS, Unitatea de legături simple
ULM, Unitatea de legături multiple
MB 0 ... MB 3 ... Unităţi de bandă magnetică
MB 0, MD 1 ... Unităţi de disc magnetic
CTC, cuplor pentru teletransmie (funcţionare în mod caracter)
CTQM, Cuplor pentru teletransmisie (funcţionare în mod mesaj)

1.16 *Unitatea de comandă*, extrage instrucţiunile programului din memoria centrală şi le analizează. Pentru aceasta ea cuprinde două registre:

— *registrul contor de instrucţiuni*, care conţine adresa noii instrucţiuni de executat;

— *registrul instrucţiune*, care conţine instrucţiunea de executat.

În general acest din urmă registru cuprinde două părţi: o parte în care se memorează codul instrucţiunii şi o parte în care se memorează adresa operandului (datelor).

După analiza fiecărei instrucţiuni sînt lansate comenzile către unităţile calculatorului (memorie centrală; unitate aritmetică şi logică etc.) pentru a executa diferite faze ale instrucţiunii.

În principiu, o instrucţiune care are efect asupra unui operand ce se găseşte în memoria centrală se derulează în următoarele trei faze:

— căutarea şi analiza instrucţiunii, pe baza conţinutului registrului contor de instrucţiuni (conţinutul acestui registru indică numărul de ordine al instrucţiunii de citit din memoria centrală);

— căutarea sau aranjarea operandului;

— pregătirea pentru instrucţiunea următoare prin adăugarea lui 1 la conţinutul registrului contor de instrucţiune.

1.17 *Unitatea aritmetică şi logică* realizează operaţii:

— *aritmetice*
- — adunare
- — scădere
- — înmulţire
- — împărţire

— *logice*
- — încarcă şi testează
- — încarcă dacă este pozitiv
- — transferă dacă este zero
- — transferă dacă este egal
- — adună dacă este plus
- — scade dacă este minus
- — stop pe eroare etc.

1.18 *Sistemul de intrări-ieşiri* cuprinde ansamblul echipamentelor care permit realizarea operaţiilor de intrare/ieşire. Controlul acestor operaţii este ierarhizat, componentele sistemului fiind grupate pe mai multe nivele de comandă în funcţie de echipamentul care participă la transferul de informaţii.

În sistemele FELIX C nivelul de comandă 1 corespunde unităţilor de schimburi multiple iar nivelul de comandă 0-unităţilor de legături şi echipamentelor periferice; componentele de nivel 1 sînt coordonate de unitatea centrală.

Unităţile de schimburi permit realizarea operaţiilor de intrare/ieşire, transferînd datele între memoria centrală şi periferice (în ambele sensuri). Ele pot fi asimilate unui calculator specializat în efectuarea operaţiilor de intrare/ieşire.

1.19 *Unităţile de legături* servesc ca interfaţă între echipamentele periferice şi unităţile de schimburi multiple.

1.20 *Echipamentele periferice* asigură înregistrarea datelor pe suporturi tehnice de date și respectiv citirea acestora pentru a fi supuse prelucrării. Un echipament periferic execută două tipuri de operații : poziționarea și deplasarea suportului de date și citirea/înregistrarea datelor.

1.21 Ca echipamente periferice de intrare se folosesc :

- lectorul de cartele perforate;
- lectorul de bandă de hîrtie;
- unitățile de benzi magnetice;
- unitățile de discuri magnetice;
- mașina de scris;
- unitățile de afișaj;
- terminale pentru teleprelucrare.

1.22 Ca echipamente periferice de ieșire se folosesc:

- imprimata;
- perforatorul automat de cartele;
- unitățile de benzi magnetce;
- unități de discuri magnetice;
- mașina de scris;
- unitățile de afișaj;
- terminale pentru teleprelucrare

1.23 Î. O configurație de calcul FELIX C cuprinde :

. . . .
. . . .
. . . .

R. — memoria centrală
— unitatea centrală
— unități de schimburi multiple
— unități de legături
— unități periferice

1.24 Î. Scrieți principalele suporturi tehnice de date: . . . .
. . . .
R. Cartela perforată, banda magnetică, banda de hîrtie, discul magnetic, hîrtia de imprimantă.

1.25 Î. Suporturile tehnice de date se folosesc pentru memorarea . . . .
. . . .
și . . . .
R. datelor, programelor

1.26 Î. Ansamblul echipamentelor cuplate direct la calculator formează
. . . .

R. Configurația de calcul

1.27 Î. La calculatorul electronic FELIX C-256, memoria centrală poate avea . . . . . blocuri, fiecare bloc avînd capacitatea de . . . . K. Un K este format din . . . . . octeți.
R. 4 ; 64 ; 1 024

1.28 Î. Programul și datele care se prelucrează trebuie să se găsească în timpul execuției în . . . .
R. memoria centrală

1.29 Î. Configurația de calcul cuprinde ca echipamente de intrare
a) . . . ., ca echipamente de ieșire
b) . . . ., iar ca echipamente de intrare/ieșire
c) . . . .
R. a) lectorul de cartele perforate, lectorul de bandă de hîrtie;
b) imprimanta, perforatorul automat de cartele;
c) unitățile de benzi magnetice, unitățile de discuri magnetice, mașina de scris, terminale pentru teleprelucrare, unitățile de afișaj.
Răspunsurile sînt corecte?
DA → 1.30.
NU ↶ 1.1.

1.30 Î. Știți ce sînt limbajele de programare?
R. DA → 1.55
NU → citiți paragrafele următoare.

1.31 Așa cum s-a prezentat, calculatorul electronic este constituit dintr-un ansamblu de unități cuplate direct sau indirect la unitatea centrală.

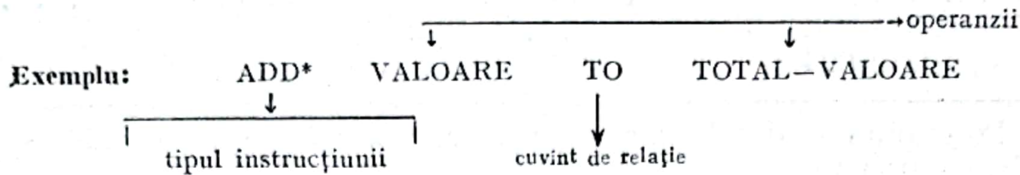
Prin construcție, unitatea centrală este prevăzută cu circuite capabile să execute anumite operații de prelucrare.

Circuitele sînt astfel concepute încît să realizeze *operații primare de prelucrare.* Pentru a se realiza în mod automat o lucrare, trebuie să se specifice prin program operațiile de prelucrare și ordinea de execuție a acestora, în conformitate cu algoritmul lucrării.

1.32 Programarea calculatoarelor implică folosirea unor *limbaje de programare.* Aceste limbaje reprezintă mijloace sau procedee care servesc la exprimarea sub formă de instrucțiuni executabile, nemijlocit de către calculator, a algoritmului de rezolvare (executare) a unei probleme. Limbajele de programare sînt sisteme convenționale de descriere a procedurilor de prelucrare automată a datelor servind ca mijloc de comunicare; ele comportă simboluri de bază (elemente primare) care asamblate în cuvinte sau expresii formează *vocabularul limbajului.* Ansamblul regulilor pe baza cărora se alcătuiesc frazele limbajului formează *gramatica limbajului.*

1.33 Un program reprezintă algoritmul de rezolvare a unei probleme, exprimat într-o suită logică de instrucțiuni.

1.34 O instrucţiune are două părţi:
— prima parte care indică codul (tipul) instrucţiunii;
— a doua parte care indică elementele (operanzii) la care se referă.



1.35 *Î.* Ansamblul comenzilor ce se dau calculatorului pentru executarea unei lucrări se numeşte . . . .
R. program

1.36 *Î.* În instrucţiunea DIVIDE** VALOARE BY 12 tipul comenzii este . . . . iar operanzii sînt . . . .
R. DIVIDE; VALOARE şi 12 (BY este cuvînt de relaţie)

1.37 Limbajul de bază al calculatoarelor electronice este *limbajul binar*:

$$L = \{0,1\}$$

O comandă (o instrucţiune calculator) se codifică printr-o combinaţie de 0 şi 1. Programele în binar sînt direct executabile de către calculator.

Unitatea de comandă analizează, una după alta, instrucţiunile programului din memoria centrală, le interpretează şi le execută.

1.38 La realizarea unui program se folosesc instrucţiuni de prelucrare (calcul, intrare/ieşire, transfer) şi de salt. Spre deosebire de instrucţiunile de prelucrare (fig. 1.4) cele de salt indică unităţii de comandă faptul că trebuie să analizeze o instrucţiune, alta decît cea care urmează în cadrul secvenţei (fig. 1.5).

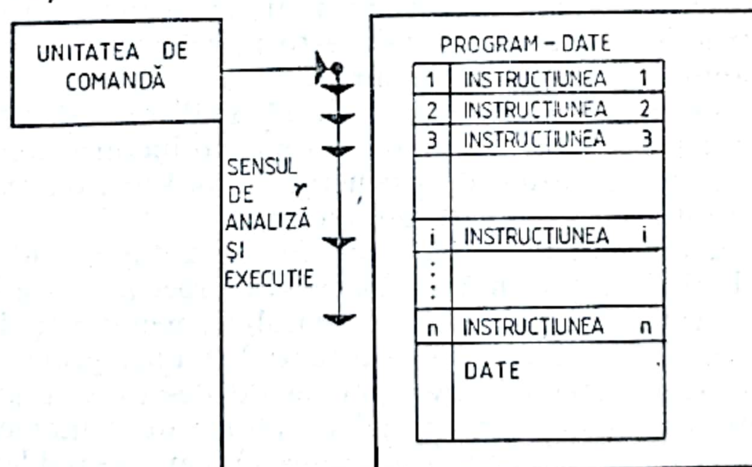


Fig. 1.4.

* ADD — adună
** DIVIDE — împarte

1.39 **Exemplu** de instrucţiune în limbajul binar

0001	0000001010	0000001001
COD		

Prin această instrucţiune se indică adunarea conţinutului de la adresa 1010 (OPERAND 1) cu cel de la adresa 1001 (OPERAND 2).

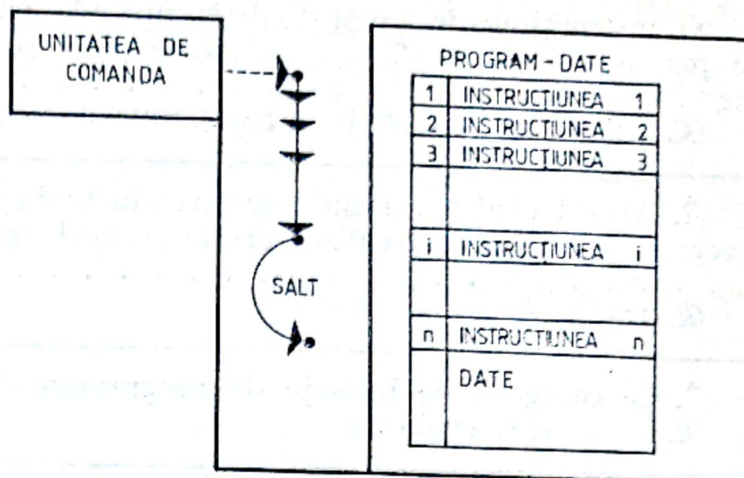


Fig. 1.5.

1.40 Programarea în limbaj binar, folosind un sistem de coduri corespunzător setului de instrucţiuni, cere timp şi efort deosebit din partea programatorului. În acest limbaj au fost programate la început calculatoarele elecronice (el se mai utilizează şi astăzi la implementarea unui sistem de calcul).

S-a impus cu necesitate ca programatorii să utilizeze limbaje mai acesibile. Aşa au apărut limbajele simbolice şi apoi limbajele evoluate. Deoarece calculatoarele electronice lucrează în limbajul de bază, programele scrise în limbaje externe *, aşa cum o să vedem în paragraful 1.48, trebuie traduse în limbajul de bază folosind programe speciale de traducere (compilatoare).

1.41 În funcţie de specificul lor, limbajele folosite în programare se împart în :
— limbaje simbolice ;
— limbaje evoluate.

1.42 *Limbajele simbolice* au la bază utilizarea unor mnemonice pentru fiecare tip de operaţie executabilă de către maşină ; calculatoarele FELIX C folosesc limbajul simbolic ASSIRIS.

Exemplu de instrucţiuni în limbaj ASSIRIS:

```
        CSECT
DATA    DATA,1,1    12,50
R2      EQU         2
ALFA    RES         1
S       LD1,R2      DATA
```

1.43 Limbajele simbolice ţin cont de particularităţile constructiv-funcţionale ale calculatorului pentru care au fost concepute. La redactarea programului se porneşte de la algoritmul lucrării, dar se urmăreşte cu deosebită atenţie modul practic de derulare a prelucrării. De aceea se spune că ele sînt orientate pe maşină.

1.44 *Ə.* În perioada de început, calculatoarele electronice se programau în limbaj . . . .

* simbolice sau evoluate

O instrucțiune în acest limbaj cuprinde partea privind . . . . . . și partea de . . . . Ambele părți se scriu printr-o succesiune de . . . .

R. Binar; tipul (codul) instrucțiunii; adresă; 0 și 1.

1.45 *Î*. Atunci cînd programele se scriu în limbaje externe pentru a obține programul în limbajul calculatorului se realizează operația de . . . .

R. traducere

1.46 *Î*. Ce categorii de limbaje de programare (externe) cunoașteți?

R. simbolice; evoluate.

1.47 *Î*. Care este limbajul simbolic specific calculatorului FELIX C-256?

R. ASSIRIS

1.48 Alături de limbajele simbolice utilizate mai mult pentru realizarea programelor de bază ale calculatorului, au fost realizate limbaje cu orientare către programator (către limbajul acestuia), fapt pentru care au fost numite evoluate.

1.49 Printre *limbajele evoluate* menționăm:

a — *Limbajul FORTRAN*, destinat îndeosebi pentru programarea lucrărilor tehnico-științifice (date puține cu calcule complexe);

b — *Limbajul COBOL*, destinat pentru prelucrarea datelor din domeniul economic (volum mare de date organizate în fișiere și calcule puține);

c — *Limbajul PL 1*, care prin construcție și posibilitățile ce le oferă îmbină unitar și complet avantajele limbajelor COBOL și FORTRAN;

d — *Limbajul BASIC* destinat îndeosebi pentru realizarea de lucrări tehnico-științifice în regim conversațional.

Ultimele extinderi ale acestui limbaj au urmărit realizarea unor facilități în vederea prelucrării datelor organizate în fișiere. El cunoaște cea mai largă utilizare în aplicațiile de teleprelucrare în timp real.

1.50 Limbajele evoluate au fost elaborate cu scopul de a fi utilizabile pe toate calculatoarele și de a ușura munca programatorului, permițîndu-i acestuia să urmărească mai mult specificul lucrării și mai puțin maniera de rezolvare pe un anumit calculator electronic. Prin urmare limbajele evoluate sînt universale.

1.51 *Î*. Să recapitulăm pentru a vă controla dacă aceste probleme sînt clare.

Limbajele specifice unui anumit tip de calculator se numesc . . . . iar cele care sînt independente de caracteristicile constructiv-funcționale ale calculatorului se numesc limbaje . . . .

Din această ultimă categorie putem enumera limbajele: . . . .

R. simbolice; evoluate; FORTRAN, COBOL, PL 1, BASIC

1.52 Î. Pentru calcule științifice se utilizeză limbajele . . . . iar pentru prelucrarea datelor organizate în fișiere limbajele . . . .

R. FORTRAN, PL1, BASIC; COBOL, PL1, BASIC,

1.53 Î. Orice calculator funcționează pe bază de . . . . care în momentul prelucrării trebuie să se găsească (parțial sau total) în . . . .

R. program
memoria centrală

1.54 Î. Programul direct executabil de către calculator este în limbaj . . . . iar programatorul scrie programul în limbaj . . . .

Aducerea lui în format direct executabil cere operația de . . . .

R. binar
extern
traducere

1.55 Traducerea automată a programelor din forma externă (SURSA) în forma internă (OBIECT) se realizează de către un program special numit *compilator* iar operația se numește *compilare* (fig. 1.6).

1.56 Compilarea presupune (fig. 1.7):

— *analiza lexicală* sau delimitarea elementelor constitutive ale programului scris (cuvintele);

— *analiza sintactică*, care permite verificarea modului în care au fost respectate regulile gramaticale ale limbajului, adică dacă cuvintele mai sus delimitate au fost combinate corect în formularea unor expresii.

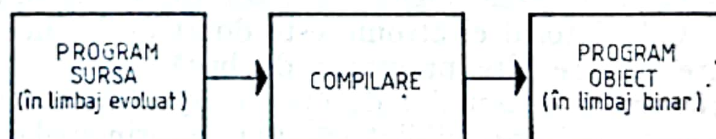


Fig. 1.6.

Orice eroare întîlnită în cele două etape este semnalată programatorului. Nu se poate trece în faza următoare decît dacă programul este corect sintactic (sau cu erori minore ce prin convenție se admit).

— *generarea programului obiect*, înseamnă

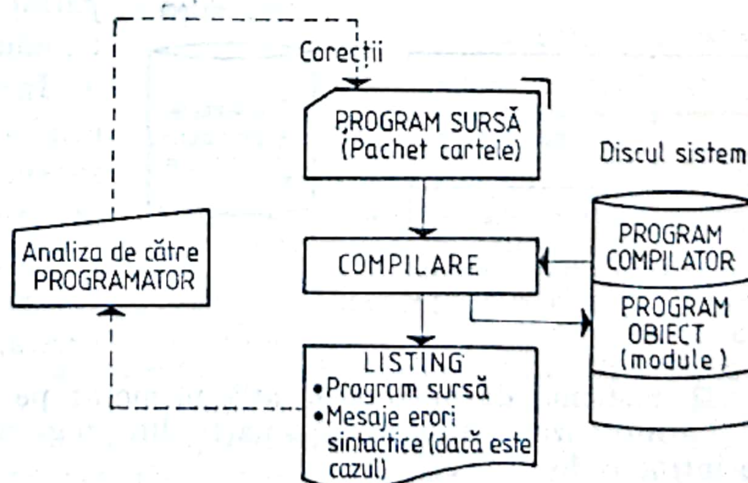


Fig. 1.7.

obținerea automată a programului (modulelor acestuia) în limbaj mașină.

1.57 *Î.* Compilatorul este un . . . . iar fiecare limbaj evoluat are un compilator asociat. Compilatorul trece programul din forma . . . . în forma . . . .

R. 1. program special
sursă
obiect

1.58 Așa cum rezultă din fig. 1.7 operația de compilare se repetă (bine înțeles după ce au fost eliminate erorile sintactice detectate) pînă cînd programul nu conține erori (sau erorile sînt admisibile).

1.59 Rezultă din paragraful 1.56 că pentru realizarea operațiilor de compilare se solicită un program special numit compilator.

1.60 Pe lîngă programul utilizatorului care este numit și *program de aplicație*, calculatorul dispune de un ansamblu de *programe de bază* care realizează anumite funcții (cum este compilarea) sau conduc activitatea întregului sistem pentru o mai rațională utilizare a componentelor sale și pentru a face ca intervenția omului să fie minimală. Prin urmare, orice sistem de calcul (sau cum se mai spune *de prelucrare automată a datelor*) se compune din :

— *componente fizice* (echipamentul), în engleză hardware;
— *componente logice* (programe), în engleză software;

Componentele logice pot fi la rîndul lor *de bază* fără de care funcționarea sistemului de calcul nu este posibilă și *de aplicație* care sînt destinate rezolvării unor probleme din diferite domenii de activitate.

Componentele logice de bază formează împreună *sistemul de operare*.

1.61 Calculatorul electronic este dotat de la început cu un *sistem de operare* care printre alte programe de bază include și compilatoarele. De obicei programele sistemului de operare sînt memorate pe o unitate de discuri magnetice într-o bibliotecă. Partea principală a sistemului de operare se introduce în memoria centrală (fig. 1.8) și formează ceea ce vom numi *partea rezidentă* sau nucleul sistemului de operare.

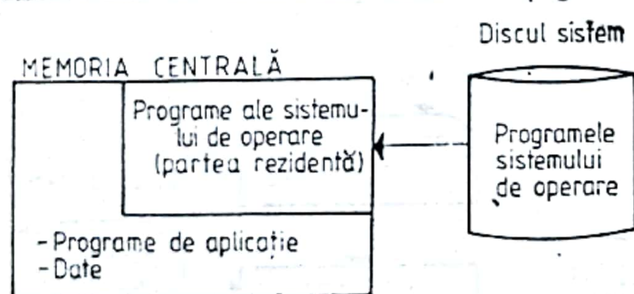


Fig. 1.8.

Întrucît sistemul de operare conduce activitatea întregului sistem de calcul, ori de cîte ori se cere executarea unei lucrări pe calculator trebuie să se realizeze mai întîi legătura cu acesta.

1.62 *Î.* Sistemul de operare se află memorat pe . . . . La inițializarea sistemului o parte din programele sistemului de operare se introduc în . . . .

R. discul sistem
memoria centrală

1.63 În timpul execuției programele de aplicație sînt însoțite de programe ale sistemului de operare. Legătura între programul de aplicație și sistemul de operare, prin care se formulează cerințele lucrării (în contul cui se realizează lucrarea, cine este programatorul, ce programe de bază solicită lucrarea, ce echipamente periferice sînt necesare executării lucrării etc.) se face cu ajutorul unui limbaj de comandă (JOB CONTROL). La sistemul FELIX C comenzile date în conformitate cu vocabularul și gramatica limbajului de comandă se perforează pe cartele cu punct.

1.64 Pentru compilarea unui program, spre exemplu, sînt necesare următoarele cartele de comandă (fig. 1.9).

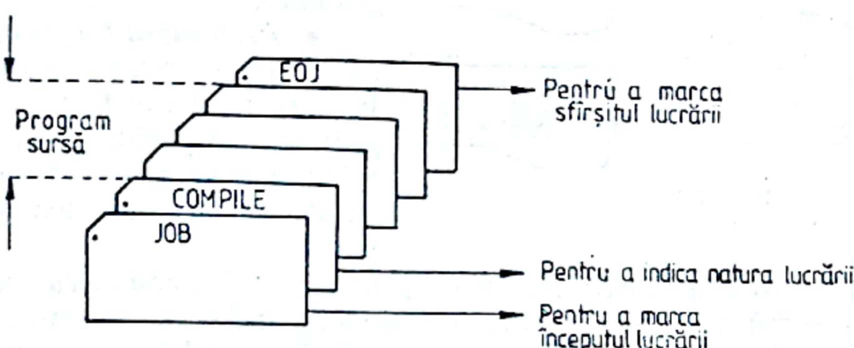


Fig. 1.9.

Prin urmare un program, pentru a fi executat de către calculator, trebuie să fie însoțit și de *cartele de comandă* care conțin instrucțiunile și informațiile adresate sistemului de operare.

1.65 *Î.* După destinația lor, programele se împart în . . . . ce intră în dotarea calculatoarelor electronice și . . . .

Acestea din urmă se întocmesc de către . . . .

R. programe ale sistemului de operare
programe de aplicații
programatorul de aplicații.

1.66 *Î.* Cum se realizează legătura dintre programele de aplicații și sistemul de operare?

. . . .

R. prin intermediul comenzilor din cartelele de comandă.

1.67 Pentru a rezolva o problemă cu ajutorul calculatorului electronic trebuie mai întîi construit programul, deci desfășurată o activitate care are ca obiectiv proiectarea, realizarea și verificarea lui, astfel spus să se desfășoare o activitate de programare.

1.68 Complexitatea acestei activități impune parcurgerea mai multor etape.

1.69 • *Studierea problemei pe baza dosarului de analiză*, care conține o descriere generală a lucrărilor de realizat pe calculator. În dosarul de analiză

sînt descrise intrările, ieşirile şi algoritmul de calcul (fig. 1.10). Studiul problemei din dosarul de analiză permite cunoaşterea acesteia de către programator.

1.70 • *Proiectarea programului,* etapă în care problema se descompune mai întîi în module (grupuri de operaţii executate în condiţii similare) şi apoi în operaţii elementare. Această etapă implică utilizarea unor procedee specifice pentru reprezentarea algoritmului (vezi lecţia 3).

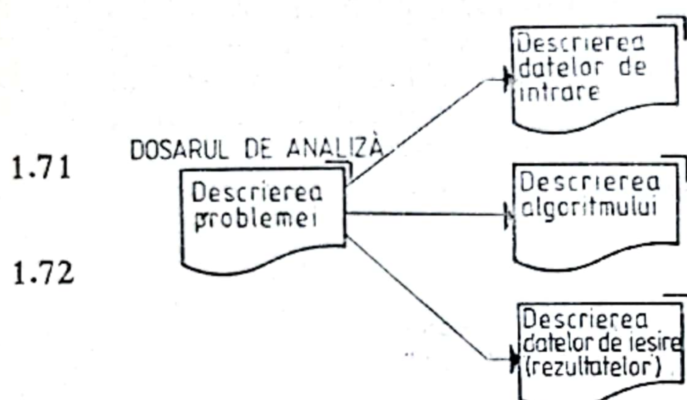


Fig. 1.10.

1.71 • *Codificarea sau scrierea programului pe formularul* de programare.*

1.72 • *Perforarea programului pe cartele* şi obţinerea programului sursă. Fiecare rînd din formularul de programare se perforează în cîte o cartelă; pachetul de cartele astfel obţinut conţine imaginea programului sursă.

1.73 • *Compilarea, editarea legăturilor, testarea şi depanarea programului.* Programul sursă (însoţit de cartelele de comandă) se compilează pentru depistarea şi eliminarea erorilor de sintaxă. În cazul în care programul nu are erori (sau are erori de mică gravitate care nu afectează rezultatele prelucrării) se trece în faza de editare a legăturilor. În această fază între modulele rezultate în urma compilării se realizează anumite legături (vezi lecţia 16) pentru a se putea trece la încărcarea programului în memoria centrală şi la testarea lui.

Pe baza unui set de date programul este în continuare testat pentru a vedea dacă rezultatul obţinut este identic cu cel dorit. În caz afirmativ programul se consideră bun şi se trece în faza de execuţie propriu-zisă (fig. 1.11) adică de realizare pe calculator a lucrării.

Pentru a creşte performanţele în prelucrare, programele care ajung în faza de execuţie se memorează pe un suport tehnic într-o *bibliotecă de programe.*

1.74 *Î.* Programul scris pe . . . . este perforat pe . . . . şi se obţine astfel un pachet de cartele cu imaginea programului sursă.

R. formularul de programare
cartele

1.75 *Î.* Prin compilare se depistează erorile de . . . . iar dacă acestea lipsesc se generează programul . . . .
. . . .

Pentru a elimina erorile de logică (semantică) se verifică . . . . prin testare, urmărind dacă rezultatele obţinute

---

* Formularul de programare este un document tipizat pe care se scriu programele.

corespund cu cele dorite (pentru aceasta se aleg anumite date de testare) Se va consulta fig. 1.11, lecțiile 18 și 19.

Rezultatul compilării constituie module în binar translatabil (BT). Prin editarea legăturilor se obține programul obiect în format imagine memorie translatabilă (IMT)

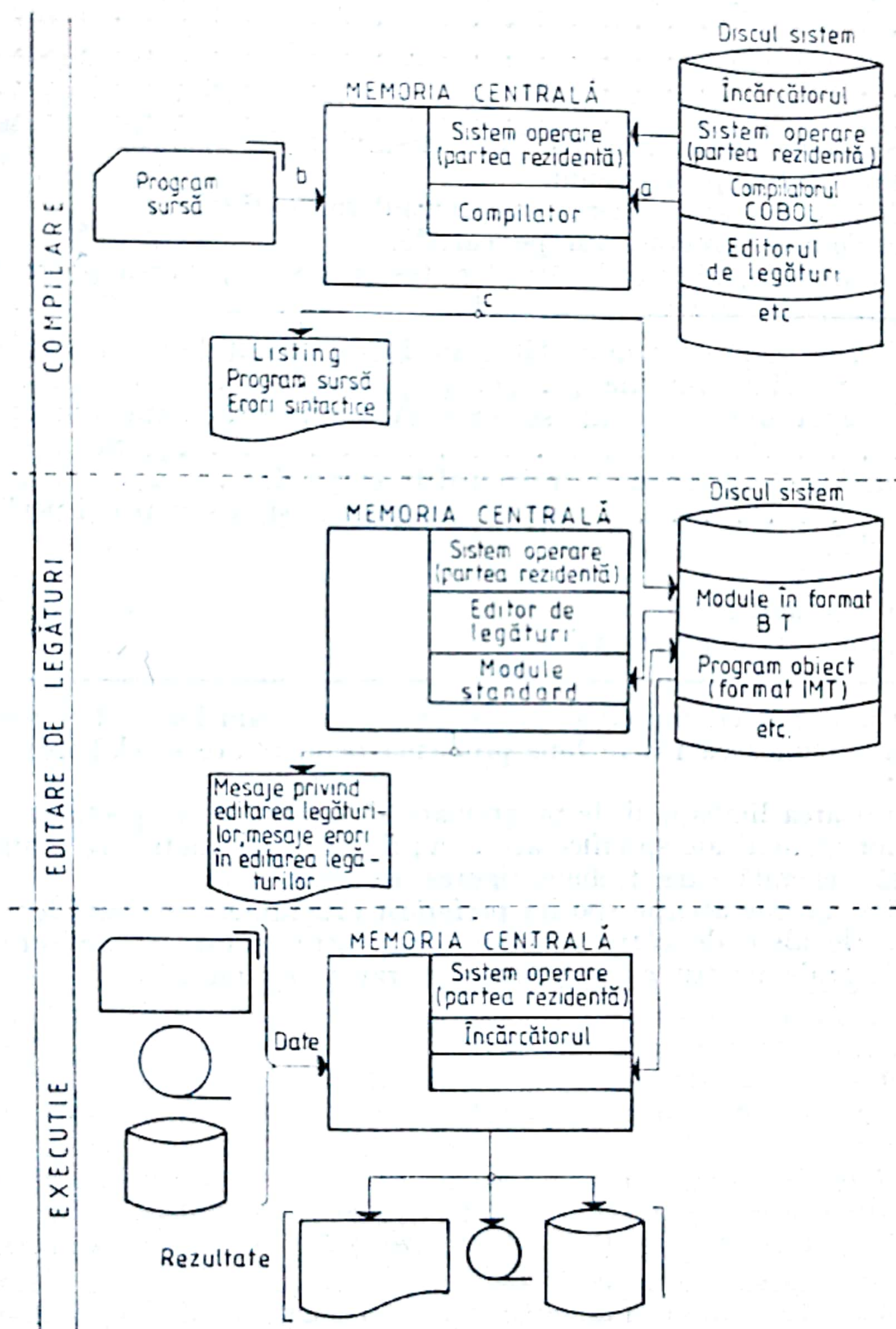


Fig. 1.11.

R. sintaxă
obiect
programul

1.76 *I.* Care sînt principalele etape pentru elaborarea programelor?
. . . .

R. Studierea problemei pe baza dosarului de analiză.
Proiectarea programului.
Codificarea sau scrierea (pe formularul COBOL).
Perforarea programului pe cartele.
Compilarea, editarea legăturilor, testarea și depanarea programului.

1.77 *I.* Generarea programului obiect are loc în condițiile în care în faza de compilare nu sînt erori de . . . .
Compilarea unui program se execută o singură dată numai dacă . . . . nu are erori.
Erorile de logică se produc în principal în etapa de . . . . dar se pot produce și la scrierea și perforarea programului.

R. sintaxă
programul sursă
proiectare a programului

1.78 Pentru a reduce timpul de elaborare a programelor este necesar:
— să se cunoască foarte bine problema pentru care se elaborează programul;
— utilizarea limbajului de programare să se facă cu respectarea strictă a regulilor gramaticale specifice acestuia; fiecare construcție are o anumită structură generală care trebuie riguros respectată;
— să se acorde atenție sporită perforării programelor pe cartele;
— datele alese de către programator pentru testare să reflecte fidel condițiile reale pentru care a fost elaborat programul.

LECŢIA 2 FIŞIER: CONCEPT, COMPONENTE, OPERAŢII, ORGANIZARE ŞI ACCES

- noţiunile de fişier, articol, cîmp
- operaţii cu fişiere
- organizarea fişierelor
  - organizarea secvenţială
  - organizarea secvenţială-indexată
  - organizarea selectivă

2.1 În procesul de prelucrare automată a datelor se pot folosi *date elementare* şi *structuri de date* (adică colecţii de date elementare care sînt în relaţie unele cu altele). Datele structurate înzestrate cu anumite operaţii formează o structură de date.

În programare distingem [14]:

- *date elementare* care pot fi:
— de tip numeric;
— de tip logic;
— de tip caracter;
— de tip pointer.
- *structuri de date* care pot fi:
— de tip tablou;
— de tip şir;
— de tip listă;
— de tip articol;
— de tip fişier.

În cele ce urmează vom prezenta pe larg structurile de date de tip fişier, dată fiind utilizarea cu prioritate a acestora în domeniul economic (ales în lucrare ca domeniu de aplicaţie).

Aşa cum am văzut în paragraful 1.2 datele, în momentul în care sînt prezentate prelucrării, trebuie să se găsească în memoria centrală. În caz contrar trebuie date comenzile necesare aducerii lor de pe suporturile tehnice. Pentru a fi utilizate direct de către beneficiari sau pentru a servi şi în alte prelucrări, rezultatele prelucrării se înregistrează, de asemenea, pe suporturi tehnice.

Întrucît datele nu pot fi înmagazinate pe un termen nelimitat în memoria centrală, există în permanenţă o legătură în ambele sensuri între memoria centrală şi cea auxiliară (suporturi tehnice), ca în fig. 2.1.

2.2 Memoria centrală are o capacitate limitată şi prin urmare, în majoritatea cazurilor, datele problemei cînd sînt în volum mare nu pot fi memorate în totalitate deodată. Acestea sînt introduse în mod succesiv de pe suporturi

tehnice pe baza operaţiilor de intrare/ieşire cu folosirea echipamentelor periferice adecvate.

2.3 După cum se ştie domeniului economic îi este specific un volum mare de date, cu particularitatea că sînt uniform structurate. Documentele primare (fig. 2.2), fişele, situaţiile etc., cuprind succesiuni de date care au, în general, aceleaşi structuri.

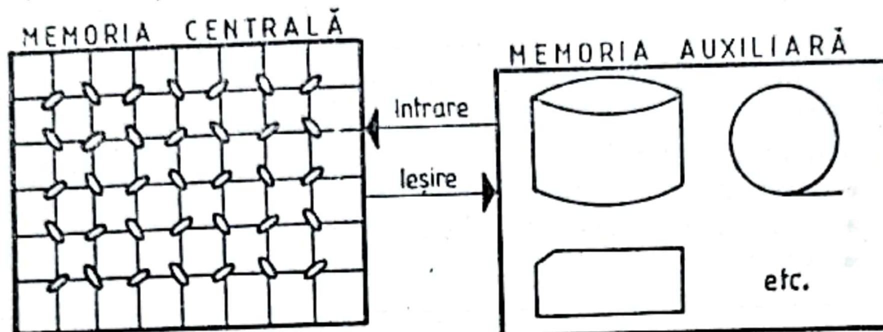


Fig. 2.1.

Cod operaţie	Unitatea	Predare Bon de transfer nr. .... Restituire

MAG. Secţia predătoare	MAG. Secţia primitoare	Număr comandă	Cont debitor	Cont creditor

DENUMIREA VALORILOR MATERIALE	CODUL	UM	CANTITATEA EFECTIVĂ	PREŢ UNITAR	VALOAREA
— — — —	— —	—	— — —	— — —	— — — —
Data şi semnătura					

Fig. 2.2.

Se impune trecerea datelor pe suporturi tehnice în aşa fel încît operaţiile de intrare/ieşire să se realizeze în condiţii optime.

Organizarea datelor, care desemnează modalitatea de depunere a acestora pe suporturi tehnice, va avea în vedere particularităţile suporturilor tehnice.

2.4 *Î.* Încercaţi să vă aduceţi aminte principalele caracteristici ale suporturilor tehnice

a) Cartela perforată este un suport discontinuu, cu o capacitate . . . . de memorare. Suportul este nereutilizabil

și neadresabil direct. Viteza de acces la datele perforate în cartelă este . . . .

b) Banda magnetică este un suport . . . . cu o capacitate . . . . de memorare. Suportul este . . . . și . . . . Viteza de acces la datele înregistrate în banda magnetică este . . . .

c) Discul magnetic este un suport . . . . .cu o capacitate . . . . de memorare.

Suportul este . . . . și . . . .

Viteza de acces la datele înregistrate în discul magnetic este . . . .

R. a) — mică
— mică

b) — continu
— mare
— reutilizabil
— neadresabil direct
— relativ mică (dar mult mai mare ca la cartelă)

c) — continu
— mare
— reutilizabil
— adresabil direct
— mare

2.5 *Î.* Așa cum am menționat domeniul economic se caracterizează prin . . . . Datele, din punctul de vedere al elementelor componente, au în general . . . .

R. — volum mare de date
— structuri uniforme

2.6 Forma de organizare și de conservare a datelor o constituie *fișierul.* Acesta reprezintă un ansamblu organizat de date omogene din punctul de vedere al naturii și al criteriilor de prelucrare.

Exemple de fișiere: fișierul stocurilor de materiale, fișierul privind mișcările de materiale, fișierul personalului dintr-o intreprindere, fișierul furnizorilor etc.

Ansamblul datelor care caracterizează, total sau parțial, un anumit fenomen (o operație de intrare a unui material sau de intrare a unui mijloc fix etc.), un anumit obiect sau o anumită persoană formează, ceea ce vom numi de acum înainte, *un articol.* Un exemplu de articol se dă în figura 2.3.

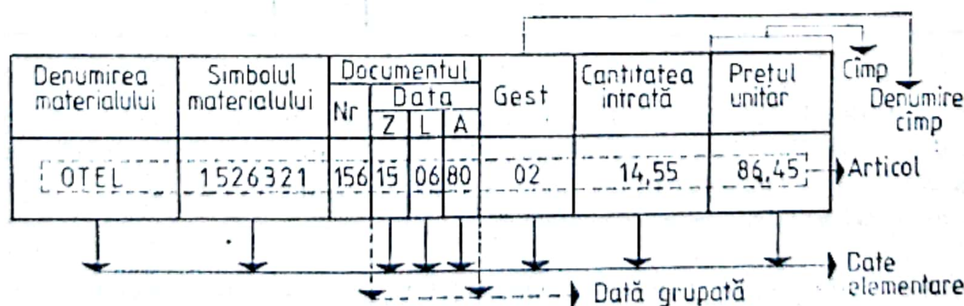


Fig. 2.3.

2.7 Unitatea elementară de organizare şi prelucrare a unui fişier de date se numeşte *articol* (sau înregistrare logică). Din acest punct de vedere fişierul se constituie dintr-o colecţie de articole. Orice entitate semantică din cadrul unui articol formează o *dată*.

În descrierea operaţiilor de prelucrare se foloseşte noţiunea generică de *cîmp*, unde data reprezintă conţinutul cîmpului.

Spre exemplu în expresia:

CANTITATE * PRET → VALOARE

se indică înmulţirea conţinutului cîmpului numit CANTITATE cu cel al cîmpului numit PRET şi depunerea rezultatului în cîmpul VALOARE:

CANTITATE		PRET		VALOARE
2 000	*	1 000	=	2 000 000

În practică prin nume cîmp se înţelege şi numele zonei de memorie la care sînt înregistrate datele.

Modul de dispunere a datelor în cadrul unui articol determină structura acestuia.

2.8 Din punct de vedere al structurii datelor dintr-un articol distingem:
— *date grupate* ce se compun din alte date (elementare sau grupate);
— *date elementare*.

În funcţie de natura caracterelor din care se compun datele pot fi:
— numerice;
— alfanumerice;
— alfabetice.

Corespunzător datelor elementare şi grupate distingem cîmpuri elementare şi respectiv cîmpuri grup.

Exemplu:

DOCUMENT			
NR.	DATA—DOC		
	ZI	LUNA	AN

GESTIUNE

unde:

— cîmpul DOCUMENT este de tip grup şi este compus din cîmpul elementar NR şi din cîmpul grup DATA—DOC, care la rîndul lui este compus din cîmpurile elementare ZI, LUNA şi AN; fiecare cîmp poartă un nume corespunzător numelui de dată:

— cîmpul GESTIUNE este elementar.

2.9 Într-un fişier se memorează articole care descriu aceleaşi entităţi (persoane, operaţii, obiecte etc); spunem că fişierul conţine o colecţie de articole omogene.

2.10 *Î.* În condițiile în care datele de supus prelucrării sînt în volum mare, ele se organizează în . . . .
Un fișier se definește ca fiind . . . .
Din punct de vedere al structurii datelor dintr-un articol se disting două categorii de date: . . . . și . . . . .
R. — fișiere
— un ansamblu organizat de date omogene din punctul de vedere al naturii și al criteriilor de prelucrare
— grupate
— elementare

2.11 În general articolele se diferențiază în funcție de valorile unui cîmp ce reprezintă cheia articolului.

Exemple.: — în cadrul fișierului PERSONAL cîmpul NUMĂR—MARCĂ constituie cheia de identificare a articolelor; fiecare persoană are un număr de marcă unic;
— în cadrul fișierului STOCURI—MATERIALE cîmpul COD—MATERIAL constituie cheia de identificare a articolelor; fiecare fel de material are un cod unic; etc.

2.12 Fișierele de date sînt memorate, pentru a fi supuse prelucrării, pe suporturi tehnice; disc magnetic, bandă magnetică etc.
Memorarea fișierelor pe suporturi tehnice se realizează sub forma *înregistrărilor fizice*: unul sau mai multe articole (în funcție de suport) sînt memorate pe cuprinsul unei înregistrări fizice.

2.13 O înregistrare fizică poate fi formată din mai multe *înregistrări logice* (*articole*) dar poate conține și una sau numai o parte a unei înregistrări logice.

2.14 Operațiile de intrare/ieșire se realizează la nivel de înregistrare fizică (bloc).
Prin înregistrare fizică înțelegem ansamblul de date care face obiectul transferului *suport tehnic — memoria centrală* la o operație de intrare-ieșire.
În general, la fișierele pe suporturi nereutilizabile articolul (înregistrarea logică) se identifică cu înregistrarea fizică; gruparea mai multor

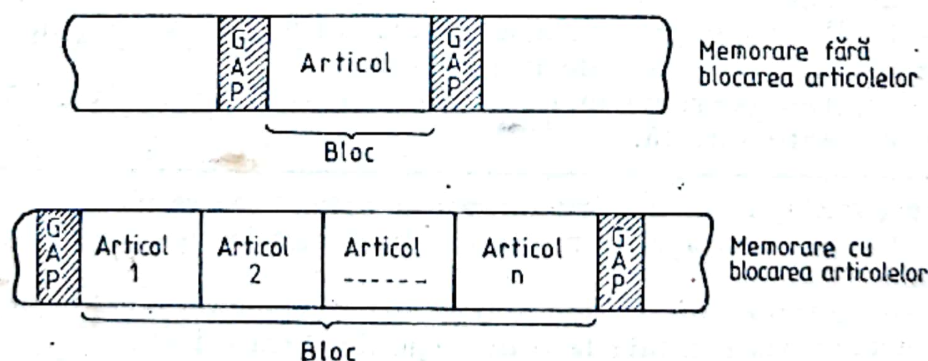


Fig. 2.4.

articole într-o înregistrare fizică este posibilă numai la suporturile magnetice. Înregistrarea fizică se mai numește și bloc (fig. 2.4).

2.15 În timp ce programatorul lucrează la nivel de articol (înregistrare logică) operațiile de intrare/ieșire se realizează la nivel de înregistrare fizică

2.16 *Î*. Cartela obişnuită are un număr de . . . . coloane iar un rînd la imprimantă are . . . . diviziuni (plus octetul de salt).

R. 80
132

2.17 Legătura între înregistrarea logică şi înregistrarea fizică este realizată în mod automat de către Sistemul de Gestiune a Fişierelor (S.G.F.), care este un program al sistemului de operare (fig. 2.5). Cînd programatorul lucrează la nivel de articol spunem că fişierul are organizare definită, iar cînd lucrează la nivel de înregistrare fizică spunem că el are organizare nedefinită.

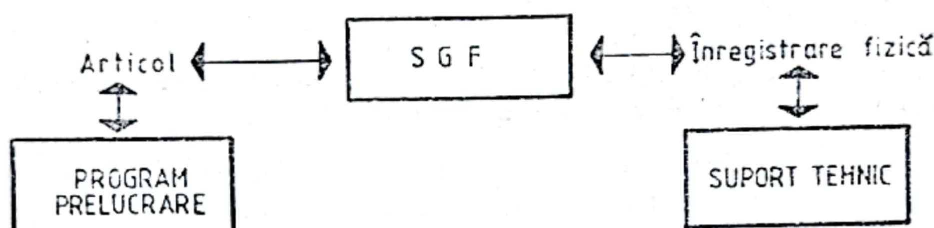


Fig. 2.5.

2.18 *Î*. Dacă o înregistrare fizică conţine mai multe . . . . , spunem că acestea sînt blocate. Blocarea articolelor este posibilă pe suporturile . . . .

R. — articole (înregistrări logice)
— magnetice

2.19 *Î*. Rezultă că un bloc este constituit din unul sau mai multe . . . . . La banda magnetică blocurile sînt delimitate prin . . . .

R. — articole
— GAP-uri (spaţii pentru a permite atingerea vitezei necesare în vederea realizării operaţiilor de intrare-ieşire)

La disc între blocuri nu mai sînt necesare GAP-urile, datorită posibilităţii de adresare directă.

2.20 Dintre avantajele memorării blocate a articolelor reţinem:

— creşterea capacităţii de memorare (mai ales la fişierele pe suporturi magnetice;

— creşterea vitezei de prelucrare prin introducerea unui volum mai mare de date în memoria centrală la o operaţie de intrare/ieşire.

2.21 Pentru a creşte performanţele sistemelor de prelucrare automată a datelor între zona de memorie afectată înregistrării articolului în vederea prelucrării (sau rezultat din prelucrare) şi zona de suport tehnic se interpune o altă zonă de memorie numită şi *zonă tampon*. În zona tampon sînt memorate înregistrările fizice ale fişierului după citirea lor de pe suport tehnic (sau înainte de a fi înregistrate pe suport tehnic).

Fiecărui fişier folosit în prelucrare i se asociază una sau mai multe zone tampon. În cazul cînd se afectează mai multe zone tampon se realizează o economie de timp, întrucît în timp ce unitatea de schimburi multiple realizează transferul unei înregistrări fizice, unitatea centrală poate realiza operaţii de prelucrare asupra înregistrării fizice transferate anterior.

Memoria afectată unui program se structurează din punct de vedere al utilizării ei în: zone tampon; zone articol şi *zone de prelucrare* (unde sînt memorate datele care participă în procesul de prelucrare, altele decît cele organizate în fişiere).

2.22 *Î*. Fişiere cu articole neblocate pot fi organizate pe suporturile
a) . . . .
în timp ce fişiere cu articole blocate pot fi organizate numai pe suporturile
b) . . . .

R. a) cartelă, bandă magnetică, disc magnetic, imprimantă
b) bandă magnetică, disc magnetic

2.23 În funcţie de lungimea pe care o au, articolele dintr-un fişier pot fi:
— de lungime fixă (F), cînd toate articolele au aceeaşi lungime;
— de lungime variabilă (V), cînd lungimea diferă de la un articol la altul;
— de lungime nedefinită (U), cînd lungimea diferă de la un articol la altul şi nu este asociată articolelor.

2.24 *Î*. Extinderea memoriei centrale se realizează prin . . . . care este organizată pe . . . .

R. — memoria auxiliară
— suporturile tehnice de date.

2.25 *Î*. În timpul execuţiei, în memoria calculatorului electronic coexistă . . . .

R. — programul de aplicaţie
— programele sistemului de operare rezidente în memorie
— datele în curs de prelucrare.

2.26 *Î*. Fişierul este format din . . . . Datele care caracterizează total sau parţial un fenomen, o persoană, un obiect pot fi . . . . şi . . . . iar din punctul de vedere al naturii . . . .

R. — articole
— grupate
— elementare
— numerice, alfanumerice, alfabetice.

2.27 *Î*. Fişierele pot avea organizare . . . . sau . . . .

R. — definită
— nedefinită

2.28 Î. Articolele unui fişier pot fi de lungime . . . .
R. fixă, variabilă sau nedefinită.

2.29

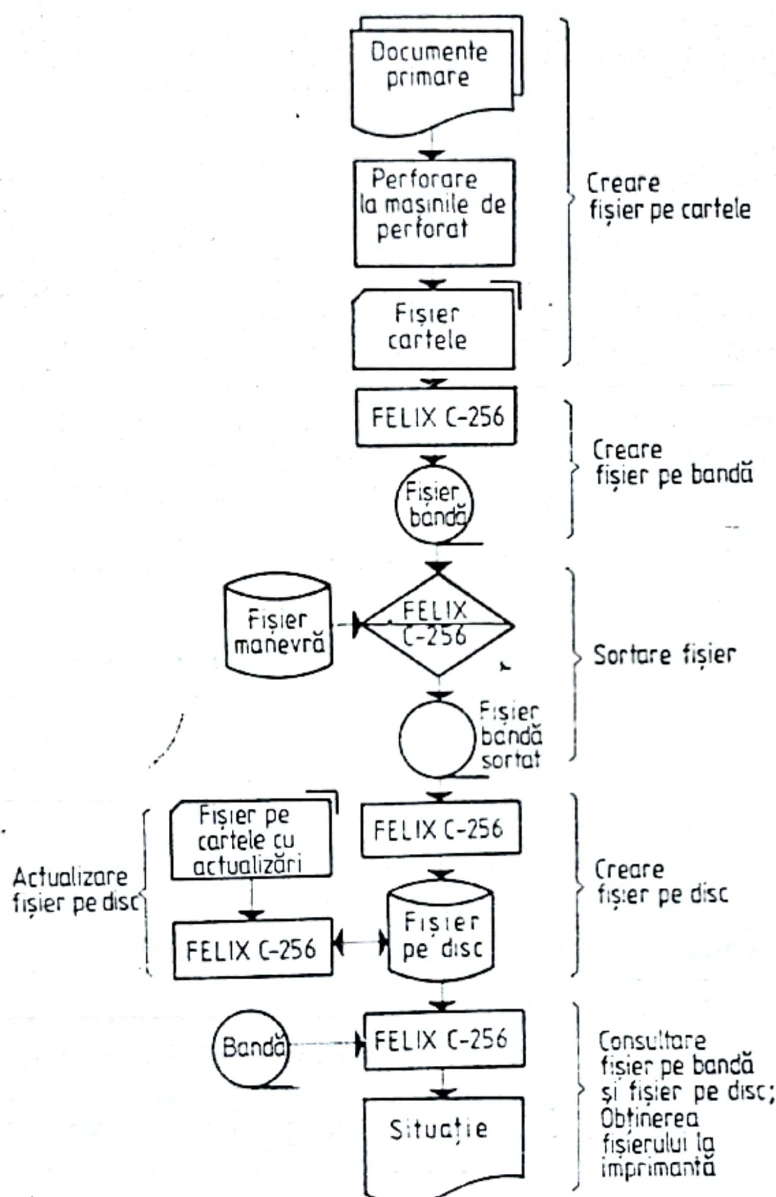


Fig. 2.6.

Un fişier de date poate fi supus următoarelor operaţii de prelucrare** (fig. 2.6) :

— *creare*, care constă în memorarea articolelor (prin perforare sau magnetizare) pe un suport tehnic;

— *sortare*, care constă în dispunerea articolelor în ordinea crescătoare sau descrescătoare a valorilor luate de un anumit cîmp * care constituie cheia de sortare;

— *consultare*, care constă în citirea articolelor din fişier în vederea prelucrării;

— *actualizare*, care constă în adăugarea de articole, modificarea conţinutului unor articole sau ştergerea unor articole.

2.30 Î. Principalele operaţii cu fişierele sînt :
. . . .
. . . .

R. — sortarea
— consultarea
— actualizarea
— crearea.

Răspunsurile din acest paragraf au fost corecte?
DA → paragraful 2.31.
NU ↶ paragraful 2.1.

* În operaţiile de sortare se pot folosi mai multe chei de sortare (vezi lecţia 11).

** În prelucrare se realizează şi operaţii ajutătoare de lucru cu fişiere cum ar fi: copiere, salvare, restaurare, ventilare şi regrupare.

2.31 *Î.* Fişierul este constituit din . . . . În condiţiile lucrului la nivel articol fişierele au organizare . . . .

R. — articole
— definită

2.32 Organizarea definită poate fi:
— secvenţială;
— secvenţială-indexată;
— selectivă.

2.33 În cazul *organizării secvenţiale*, articolele sînt dispuse în mod succesiv respectînd ordinea de apariţie.

2.34 *Î.* Această metodă este posibilă pe toate suporturile tehnice de date, adică: . . . .

R. cartelă, bandă perforată, bandă magnetică, disc magnetic, imprimantă.

Accesul la articolele unui fişier organizat după această metodă este secvenţial, ceea ce presupune parcurgerea suportului pînă se ajunge la locul de memorare al celui căutat (fig. 2.7 a).

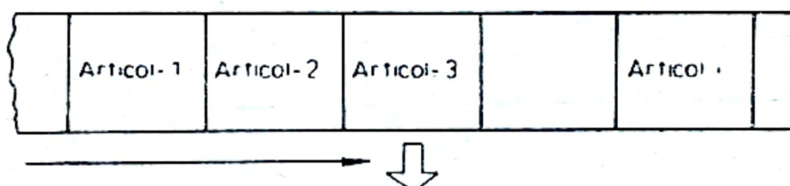


Fig. 2.7. a.

2.35 *Î.* Această metodă de organizare este singura posibilă pe suporturile neadresabile direct, adică . . . .

R. cartelă, bandă perforată, bandă magnetică, imprimantă.

2.36 *Organizarea secvenţială-indexată* este posibilă pe suporturi adresabile direct, cum ar fi spre exemplu discul magnetic. Articolele unui fişier cu organizare secvenţială-indexată sînt identificabile în mod unitar în funcţie de valorile luate de un anumit cîmp numit şi *cîmp cheie*. Înregistrarea articolelor în fişier se face în mod secvenţial, în ordinea crescătoare a valorilor luate de cîmpul cheie; în acelaşi timp ele sînt repertoriate într-o tabelă numită şi tabelă de indecşi.

Un index cuprinde, în general, două cîmpuri al căror conţinut desemnează *valoarea cheii* şi *adresa de suport* la care este memorat articolul respectiv.

Cum organizarea cîte unui index pentru fiecare articol al fişierului nu ar reduce prea mult timpul necesar pentru executarea operaţiilor de intrare/ieşire, sistemul de gestiune a fişierelor constituie cîte un index pe

grupe de articole (numărul de articole dintr-o grupă este stabilit de către S.G.F.) după structura :

— valoarea maximă luată de cîmpul cheie, în cadrul grupei de articole ;
— adresa la care s-a început memorarea articolelor din grupa respectivă.

Astfel, pentru o grupă de articole, în tabel, se memorează un index compus din valoarea cheii maxime și adresa paginii, cilindrului sau volumului.

Împărțirea fișierului în grupe de articole este asigurată în mod automat de către S.G.F. iar articolele sînt înregistrate secvențial pe suport și repertoriate într-o tabelă de indecși (fig. 2.7 b).

0	Articol 1120	Articol 1121	Articol 1123	Articol 1125	Articol 1130
1	Articol 1132	Articol 1133	Articol 1134	Articol 1139	Articol 1140
2	Articol 1142	Articol 1147	Articol 1152	Articol 1153	Articol 1155
3	Articol 1156	Articol 1160	Articol 1161	Articol 1162	Articol 1163
4	Articol 1165	Articol 1166	Articol 1167	Articol 1169	Articol 1170

Partea principală

Sector	
0	1130
1	1140
2	1155
3	1163
4	1170

Tabelul de indecși

Partea de depășire

Fig. 2.7 b.

Tabelele cu indecși se constituie la crearea fișierului. Pentru fiecare tip de unitate adresabilă (volum, cilindru, pagină) ocupată de fișier se pot constitui două tipuri de tabele :

— *tabela detaliată* în care se trece adresa tipului de unitate și valoarea cheii ultimului articol memorat în cadrul unității ;

— *tabela rezumat*, în care se trece adresa fiecărei pagini ocupate de tabela detaliată și valoarea cea mai mare a cheii memorate în pagina respectivă.

2.37 *Î.* Articolele fișierului se deosebesc între ele prin valorile luate de un anumit cîmp numit . . . .

R. cîmp cheie.

2.38 *Î.* În fișierul stocurilor de valori materiale, articolele se deosebesc prin . . . .

R. codul materialului.

2.39 Articolele înainte de a fi memorate într-un fișier cu organizare secvențială-indexată sînt sortate în ordinea crescătoare a valorilor luate de cheia de identificare.

2.40 Accesul la articolele din fişier se poate realiza:

— *secvențial*, cînd sînt citite toate articolele care preced articolul respectiv;

— *direct* *), cînd se consultă mai întîi tabelul cu indecşi pentru a determina grupa din care face parte articolul şi respectiv adresa de memorare a grupei; în cadrul grupei accesul are loc în mod secvențial.

Din punct de vedere fizic, un fişier cu organizare secvențială-indexată este constituit din 3 zone;

— *zona tabelelor cu indecşi*;

— *zona principală*, în care sînt memorate articolele la crearea fişierului;

— *zona de depăşire*, în care sînt memorate articolele ce se adaugă, dacă ele nu mai pot fi inserate în cadrul grupelor din zona principială. Printr-un sistem de legături se menține secvența logică a articolelor fişierului.

2.41 *Î.* Rezultă că fişierul cu organizare secvențială-indexată e format din partea . . . . în care se memorează articolele la constituirea fişierului, partea de . . . . . . în care se memorează articolele ce se adaugă atunci cînd nu mai pot fi memorate în zona repartizată inițial grupei şi . . . .

R. — principială
— depăşire
— tabelul cu indecşi.

2.42 *Organizarea selectivă*, posibilă numai pe suporturi adresabile direct, urmăreşte ca şi metoda de organizare secvențială-indexată accesul mai rapid la date cu deosebirea că realizarea corespondenței *articol—adresă locație* pe suport tehnic se face printr-o funcție de calcul, definită explicit, numită şi *funcție de randomizare* (în ideea efectuării mai rapide a calculelor).

Zona de suport afectată fişierului este împărțită în căsuțe (zone de lungimi egale), numerotate începînd cu zero. Numărul de ordine ataşat căsuței constituie cheia de adresare (cheia reală).

Articolele diferă între ele prin valorile luate de un anumit cîmp numit cheie:

VALOARE CÎMP CHEIE → FUNCȚIE DE RANDOMIZARE → ADRESA CĂSUȚA

Funcția de randomizare desemnează de obicei calculele ce trebuie efectuate asupra valorilor luate de cîmpul cheie atît la citire cît şi la înregistrare, pentru a obține adresa căsuței în cadrul suportului.

Fişierul cu organizare selectivă se compune dintr-o colecție de grupe (clase) cu articole sinonime, împărțirea în clase făcîndu-se în baza funcției de randomizare.

Fiecare grupă se memorează într-o căsuță. Regăsirea unui articol se face pe baza valorii cheii şi a adresei căsuței de înregistrare.

Un exemplu este redat în fig. 2.8. (căsuță=casetă)

Din punct de vedere fizic suportul de memorare a fişierului cuprinde:

— *partea principală*, formată dintr-o suită de n căsuțe de aceeaşi lungime (în fiecare căsuță se memorează o grupă de articole sinonome);

---

*) Pentru realizarea accesului direct se cer precizate: cheia de indentificare (din structura articolului) şi cheia de adresare (care indică articolul ce trebuie consultat).

— *partea de depășire*, unde se memorează articolele ce nu mai pot fi înserate în partea principală (cu menținerea secvenței logicei printr-un sistem de legătură).

Calculele de randomizare se efectuează atît la constituire cît și la exploatare, ceea ce face ca accesul să nu se poată realiza decît în mod direct.

Exemplu

152	Articol	156	Articol		Caseta 0
153	Articol				Caseta 1
154	Articol				Caseta 3
151	Articol	155	Articol		Caseta 4

Fig. 2.8.

Pentru a alege metoda de organizare a fișierului trebuie avute în vedere:

— suporturile disponibile (care în general pot fi adresabile și neadresabile);

— cerințele privind accesul la date.

În principiu, datele de stare au o participare de lungă durată în procesul de prelucrare și sînt organizate după metoda secvențială-indexată sau selectivă iar cele de mișcare au o participare de mai scurtă durată și sînt organizate după metoda secvențială.

2.43 Î. Organizarea secvențială impune ordonarea articolelor crescător după o anumită caracteristică?

R. Nu

2.44 Î. Accesul la datele fișierului organizat secvențial este . . . .

R. secvențial

2.45 Î. Organizarea secvențială-indexată este posibilă pe suporturi . . . . adică pe . . . .

R. — adresabile direct
— discul magnetic.

2.46 Î. La crearea unui fișier cu organizare secvențială-indexată accesul este . . . . , la exploatare accesul este . . . .

R. — secvențial
— secvențial sau aleator (direct).

2.47 Î. La organizarea secvențială-indexată elementul care deosebește un articol de altul este . . . . iar cel care localizează articolul pe suport este . . . .

R. — cheia articolului
— adresa locației.

2.48 Î. Într-un fișier cu organizare secvențială-indexată, articolele sînt dispuse în . . . . după valorile luate de cîmpul . . . .

𝓡. — ordine crescătoare
— cheie.

2.49 𝓘. Organizarea selectivă este posibilă pe suporturi . . . . adică . . . .

𝓡. — adresabile direct
— discul magnetic

2.50 𝓘. Într-un fișier organizat selectiv articolele se deosebesc între ele prin . . . . care precede articolul în cadrul . . . . iar numărul de ordine atașat căsuței constituie . . . . Articolele care se află în aceeași căsuță se numesc . . . .

𝓡. valorile cîmpului cheie
— căsuței
— cheia reală
— sinonime.

Răspunsurile sînt corecte

DA → paragraful 2.51

NU ↶ paragraful 2.31

2.51 *Sistemul de etichete.* Pentru a recunoaște un anumit fișier pe cartele, se poate perfora în toate cartelele un cod de fișier.

2.52 Lizibilitatea datelor este asigurată la fișierele memorate pe cartele și la imprimantă.

Fișierele pe suporturi nereutilizabile nu au nevoie de informații speciale pentru identificare și protecție; spunem că la această categorie de fișiere etichetele sînt omise.

2.53 În condițiile folosirii suporturilor magnetice, pe lîngă faptul că articolele pot fi înregistrate în mod diferit (blocat și neblocat) și pot avea lungimi diferite, există și posibilitatea distrugerii lor printr-o utilizare eronată. Apare ca firească necesitatea asigurării unei protecții privind accesul la datele fișierelor memorate pe suporturi magnetice și prin urmare adăugarea unor date de control.

2.54 Memoria auxiliară pe suporturi magnetice se compune din una sau mai multe unități de volum.

Exemple de unități de volum:
- un pachet de discuri magnetice;
- o bandă magnetică.

Un fișier poate fi memorat pe o singură unitate de volum (monofișier-monovolum) sau pe mai multe unități de volum (*monofișier-multivolum*); S.G.F.-ul permite și folosirea multifișierelor (*monovolum-multifișier*). Fișierele înregistrate pe suporturi magnetice sînt însoțite de grupuri de date speciale care asigură identificarea, delimitarea și protecția privind accesul la date și care furnizează S.G.F.-ului informațiile necesare realizării legăturii automate între înregistrarea fizică și articol; aceste grupuri de date sînt numite și etichete (sau articole de tip etichetă).

Etichetele sînt grupate la începutul și sfîrșitul fișierului dacă acesta este memorat pe bandă magnetică sau într-o zonă distinctă (zona informațiilor de control) dacă acesta este memorat pe disc magnetic (fig. 2.10). În figura 2.9. se prezintă structura simplificată a unui fișier organizat secvențial pe o singură bandă magnetică.

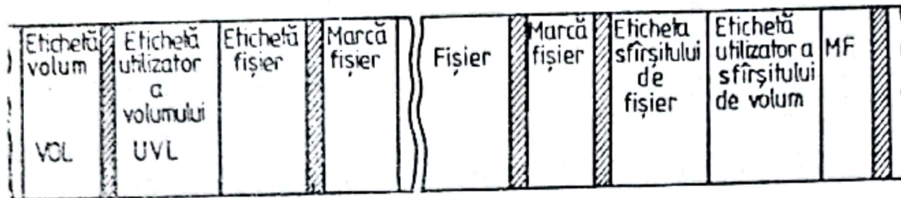


Fig. 2.9.

Etichetele pot fi:
— de început volum;
— de fișier;
— de sfîrșit volum.

2.55 În afară de etichetele create de către sistem numite și *etichete standard* pot fi create și etichete suplimentare (definite de către utilizator), numite *etichete utilizator* (numai la fișiere pe bandă magnetică).

Se face precizarea că fișierele pe bandă magnetică pot fi create și-n opțiunea cu omiterea etichetelor. La discul magnetic se utilizează numai etichete standard.

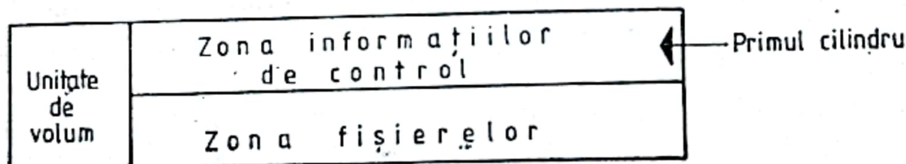


Fig. 2.10.

2.56 *Î.* Rezultă că fișierele pe suporturile cartelă și imprimantă a) . . . . cele pe bandă magnetică pot fi b) . . . . iar cele pe disc magnetic cu etichete c) . . . .

R. a) nu au etichete
b) fără etichete, cu etichete standard, cu etichete utilizator
c) standard.

2.57 *Î.* Etichetele cu o organizare impusă de sistem se numesc etichete . . . . iar cele suplimentare prevăzute de către utilizator se numesc etichete . . . .

R. — standard
— utilizator.

2.58 Accesul la articolele unui fișier se poate realiza în mod MOVE sau LOCATE.

În modul MOVE utilizatorul are acces la articole, în vederea prelucrării, prin intermediul unei zone de memorie numită *zonă-articol*. Între zona articol și *zona din suport* unde se înregistrează articolul sau de unde se citește articolul, se interpune *zona tampon*.

În modul LOCATE, utilizatorul are acces la articol direct în zona tampon (fără a mai utiliza o zonă articol).

Pentru verificarea cunoștințelor rezolvați testul 1, pagina 347.

LECȚIA a 3-a TEHNICI DE DESCRIERE ȘI REPREZENTARE A ALGORITMILOR

- algoritm și operații elementare
- structuri de bază utilizate în conceperea programelor
- reprezentarea grafică a algoritmilor : scheme logice
- limbajul pseudocod

3.1 *I.* În procesul studierii unui domeniu de aplicație în vederea prelucrării automate a datelor se întocmește *dosarul de analiză* cu problemele ce stau în atenția programatorului în scopul conceperii și realizării programelor. Dosarul de analiză conține, pentru fiecare problemă supusă studiului.

. . . .
. . . .
. . . .

R. — datele de prelucrat
— rezultatele de obținut
— algoritmul de prelucrare a datelor.

Răspunsul este corect?

DA → 3.2
NU ↰ 1.68

3.2 Un algoritm *clar*, *coerent* și *complet* constituie premiza principală de obținere a unui program fiabil.

3.3 Pentru a fi programabil un algoritm trebuie să îndeplinească cumulativ condițiile de:

— *claritate*, să exprime cu rigurozitate și fără ambiguitate operațiile de prelucrare a datelor și ordinea lor de succesiune;

— *completitudine*, să realizeze o descriere completă a procesului de prelucrare a datelor; orice omisiune are repercusiuni directe asupra calității programului și respectiv lucrării.

3.4 Descrierea algoritmilor impune cunoașterea riguroasă a problemei supusă studiului și în egală măsură însușirea bazelor teoretice ale conceperii programelor.

Un loc distinct în cadrul acestora din urmă îl ocupă operațiile primare și mărimile pe care acestea le implică.

3.5 Mărimile utilizate în descrierea algoritmului unei lucrări sînt:

— *fișier*, definit ca o colecție omogenă de articole;

— *articol*, definit ca un grup de date care cuantifică un fenomen, un obiect, un proces sau o operație etc.

— *cîmp* (variabilă), o mărime care poate lua diferite valori dintr-o mulţime de valori.

3.6 Cînd un cîmp nu poate lua decît două valori (0 şi 1 ; DA şi NU) spunem că el este de tip boolean.

3.7 *Q.* În descrierea algoritmilor se operează cu mărimile

. . . .

R. fişier, articol, cîmp (variabilă).

3.8 Prelucrarea datelor presupune o succesiune ordonată de operaţii care acţionează asupra mărimilor mai sus menţionate.

3.9 În cele ce urmează vom distinge 5 categorii de operaţii primare şi anume :

— operaţii de atribuire ;

— operaţii de calcul ;

— operaţii de decizie ;

— operaţii de intrare-ieşire ;

— operaţii de transfer a controlului.

3.10 Prin intermediul *operaţiilor de atribuire*, unui cîmp i se atribuie valoarea definită de un literal numeric, de un alt cîmp sau obţinută prin calcul.

Exemple:

V—EOF ← 0 sau V—EOF: = 0
TOTAL—VALOARE ← 0 sau TOTAL—VALOARE: = 0
CONTOR ← 65
DENUMIRE—BANDA ← DENUMIRE—CARTELA
STOC—NOU ← STOC —VECHI + INTRARI—IEŞIRI

Atribuirea unei valori este temporară, prin urmare există posibilitatea realizării unei succesiuni în timp cu alte valori.

3.11 *Operaţiile de calcul* se definesc pe mulţimea numerelor reale şi includ operaţiile de :

— adunare +

— scădere —

— înmulţire *

— împărţire /

— ridicare la putere **

3.12 Pentru operaţii complexe se folosesc parantezele, evaluarea făcîndu-se după regulile cunoscute din algebră.

3.13 *Operaţiile de decizie* servesc pentru a determina valoarea logică a unei propoziţii, care poate fi adevărată (DA) sau falsă (NU). Ele se utilizează pentru a condiţiona executarea unor operaţii sau grupuri de operaţii. La descrierea algoritmilor, operaţiile de decizie se vor constitui din valori numerice, cîmpuri, expresii folosind semnele speciale =, >, <, ≥, ≤, ≠ şi operatori logici AND, OR şi NOT.

Exemple*:

```
a) dacă NATURA–OPERAȚIE = 1
        atunci
            adună CANTITATE la TOTAL–INTRĂRI
        altfel
            adună CANTITATE la TOTAL–IEȘIRI
   sf-dacă
b) dacă STOC–EFECTIV > STOC–NORMAT
        atunci
            afișează „STOC SUPRANORMATIV"
        altfel (*)
   sf-dacă
c) dacă CANTITATE not numeric
        atunci
            afișează „CANTITATE NENUMERICĂ"
        altfel
            dacă
                CANTITATE > 1 AND < 99999
                    atunci (*)
                    altfel
                        afișează "CANTITATE VAL ER"
            sf-dacă
   sf-dacă
```

Pentru a nu face apel la enunțurile specifice unui limbaj de programare, operațiile de prelucrare (intrare/ieșire, calcul etc.) sînt desemnate prin intermediul unor cuvinte care apar cu minuscule și cursive.

În descriere se folosesc delimitatorii:

— *atunci*, care marchează începutul grupului de operații de executat cînd la testarea condiției valoarea logică a condiției este „adevărat";

— *altfel*, care marchează începutul grupului de operații de executat cînd la testarea condiției valoarea logică a condiției este „fals";

— *sf-dacă*, care marchează sfîrșitul operației de testare *dacă*.

B.14 *Operațiile de intrare/ieșire* au ca obiectiv aducerea datelor în memoria centrală în vederea prelucrării și respectiv redării rezultatelor. Ele vizează realizarea transferului de date de la memoria externă (auxiliară) în memoria centrală și invers.

B.15 *Operația de citire* a articolelor dintr-un fișier diferă, în funcție de organizarea acestuia din urmă.

Pentru fișiere cu organizare secvențială operația de citire trebuie însoțită de operația de decizie care are ca obiectiv precizarea cazului în care s-a citit cartela EOF (END OF FILE) denumită și marca EOF.

Exemplu:

* Prin (*) desemnăm mulțimea vidă de operații.

La realizarea operației de citire folosind un fișier cu organizare secvențială există două posibilități :

— să se citească un articol de supus prelucrării ;

— să se citească marca EOF.

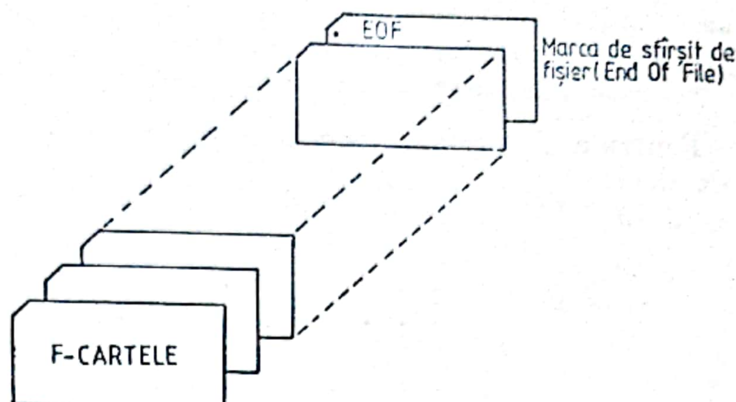


Fig. 3.1.

Pentru a delimita cazul detectării mărcii EOF, se va defini un cîmp care prin valoarea 0 precizează că mai sînt date în fișier (valoare implicită) iar prin valoarea 1 că s-a citit marca EOF.

Enunțul *citește* va fi formulat după următoarea structură generală (liniile cu * sînt linii comentarii) :

```
citește   nume-fișier
               dacă-marca-EOF
                    * operație de marcare a cazului
                    *
                    * sfîrșit de fișier.
```

Exemplu:

```
V—EOF ← 0
citește FIȘIER—CARTELE
               dacă—marca—EOF
                    1 → V—EOF.
```

Prin testarea valorii luate de cîmpul V-EOF se află dacă continuarea operațiilor de prelucrare are sens.

3.16 Pentru citirea în acces direct a articolelor ce definesc un fișier cu organizare secvențială-indexată operația de citire este însoțită de cea de decizie care vizează delimitarea cazului în care valoarea cheie a articolului desemnează un articol inexistent în fișier.

Enunțul citește în acest caz va avea următoarea structură generală :

```
citește   nume-fișier
               dacă-valoare-cheie-inexistentă
                    * operație de marcare a cazului valoare cheie
                    *
                    * articol de citit inexistent.
```

Exemplu:

```
0 → V—IVK
citește FIȘIER—SECVENȚIAL—INDEXAT
               dacă—valoare—cheie—inexistentă
                    1 → V—IVK.
```

unde:

- V-IVK, cîmp care prin valoarea 1 delimitează cazurile în care citirea articolului menţionat prin valoare cheie nu se poate realiza întrucît el nu există în fişier.

Pentru a direcţiona procesul de prelucrare se recomandă folosirea operaţiei de decizie:

dacă V-IVK = 0
atunci
. . . .
. . . .
altfel
. . . .
. . . .
sf-dacă

Continuarea procesului de prelucrare este dată de răspunsul obţinut la operaţia de decizie (fig. 3.2).

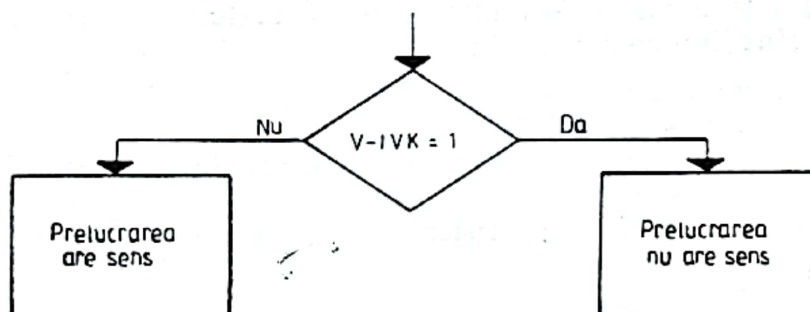


Fig. 3.2.

3.17 *Operaţia de înregistrare (scriere)* se utilizează pentru trecerea datelor din memoria centrală pe suporturi şi poate avea una din structurile:

- *scrie* nume-articol

iar pentru fişiere cu organizare secvenţială-indexată:

- *scrie* nume-articol
  *dacă-valoare-cheie-eronată**
  * operaţie de marcare a cazului
  * în care articolul nu poate fi scris.

3.18 În categoria operaţiilor de transfer a controlului includem:
— *operaţia de trimitere* (salt) într-un anumit punct al programului:
A. operaţia—1
operaţia—2
.
.
.
dacă p *atunci* *mergi* la A

* Valoarea cheie a articolului ce trebuie scris este mai mică decît valoarea cheie a articolului scris anterior.

— *operația de apel*, condiționat sau nu, al unei proceduri (fig. 3.3).

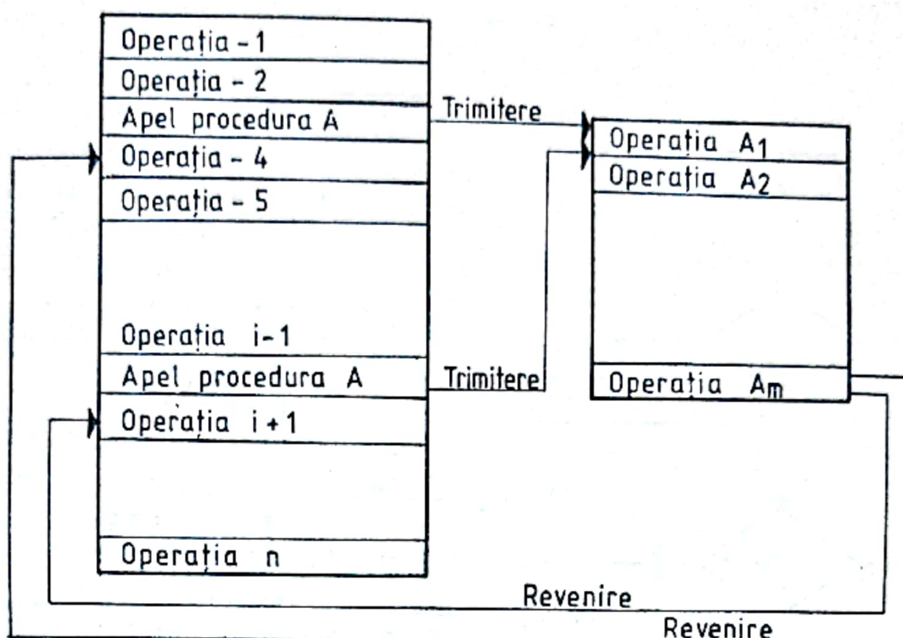


Fig. 3.3.

3.19 *Î.* Operațiile de descriere a algoritmilor se împart în . . . .

R. — atribuire
— calcul
— decizie
— intrare/ieșire
— transfer al controlului.

3.20 *Î.* Operațiile de decizie permit . . . . în funcție de valoarea logică atribuită unei propoziții care poate fi . . . .

R. direcționarea procesului de prelucrare; adevărată (DA), falsă (NU).

3.21 *Î.* Operația *citește* a datelor dintr-un fișier cu organizare secvențială numit FIȘIER-INTRĂRI-MATERIALE se prezintă astfel . . . .

ℛ (fig. 3.4)
citește articol din FIȘIER-INTRĂRI-MATERIALE
dacă-marca-EOF
1 → V-EOF.

3.22

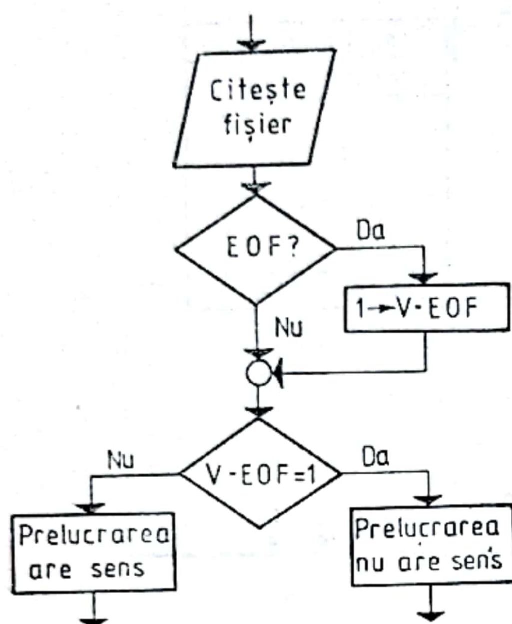


Fig. 3.4.

ℐ. Pentru detectarea cazului în care, la citirea unui fișier cu organizare secvențială, s-a detectat marca de sfîrșit de fișier se folosește . . . . care poate lua valorile . . . .

ℛ. — un cîmp (o variabilă)
— 0 sau NU
— 1 sau DA

Răspunsurile dumneavoastră sînt corecte?

DA → 3.23
NU ↶ 3.9

3.23 În descrierea algoritmului unei lucrări se folosesc structurile de bază:
— structura secvențială;
— structura alternativă;
— structura repetitivă.

Toate aceste structuri au o intrare și o ieșire.

3.24 *Structura secvențială* desemnează una sau mai multe operații executate în mod liniar (secvențial):

Exemplu (fig. 3.5.)

```
Op0: început
Op1:   CANTITATE * PRET → VALOARE
Op2:   VALOARE + TOTAL–VALOARE → TOTAL–VALOARE
Op3:   1 + NUMAR–CURENT → NUMAR–CURENT
Op4:   COD–MATERIAL → COD–MATERIAL–L
Op5:   DENUMIRE–MATERIAL  → DENUMIRE–MATERIAL–L
Op6:   CANTITATE          → CANTITATE–L
Op7:   PRET               → PRET–L
Op8:   VALOARE            → VALOARE–L
Op9:   NUMAR–CURENT       → NUMAR–CURENT–L
Op.10: sfîrșit
```

Reprezentarea grafică a structurii secvențiale este redată în fig. 3.6.

3.25 *Structura alternativă*

Reprezentarea grafică a structurii alternative poate avea una din formele din figura 3.7.

unde:

BLOC, desemnează un grup de operații executate în condiții similare.

Reluînd exemplul de mai sus, presupunem că operațiile Op1...Op9 au sens numai în condițiile în care cîmpurile CANTITATE și PRET au

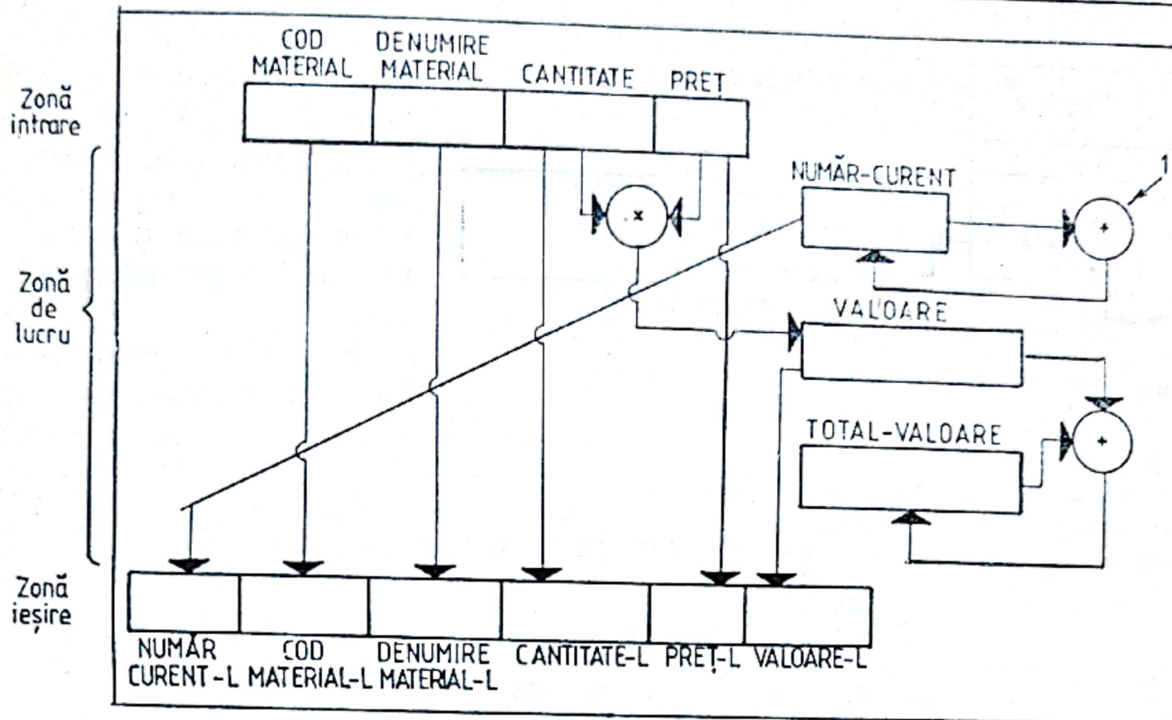


Fig. 3.5.

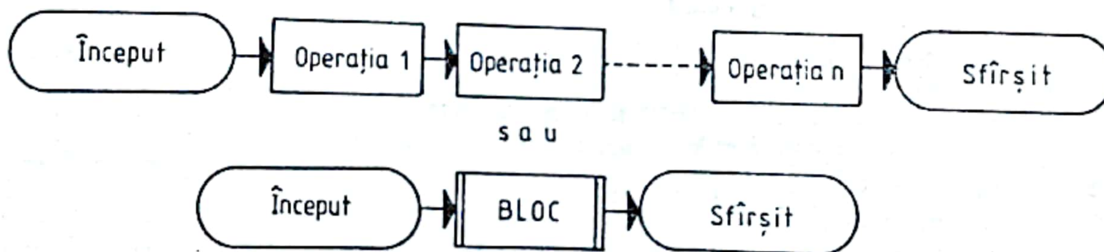


Fig. 3.6.

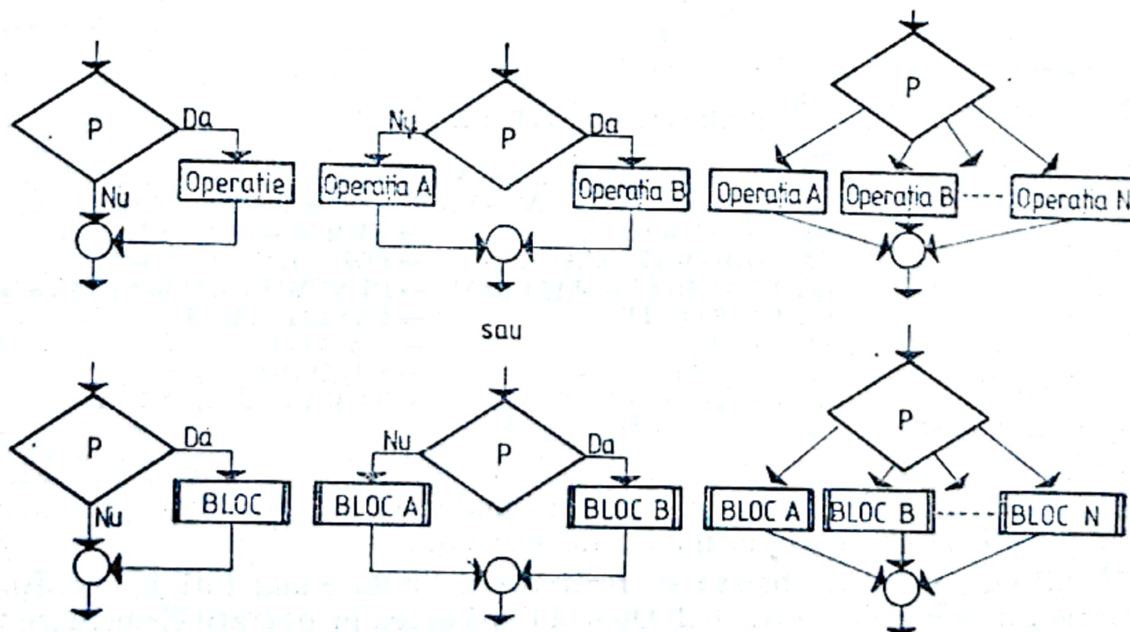


Fig. 3.7.

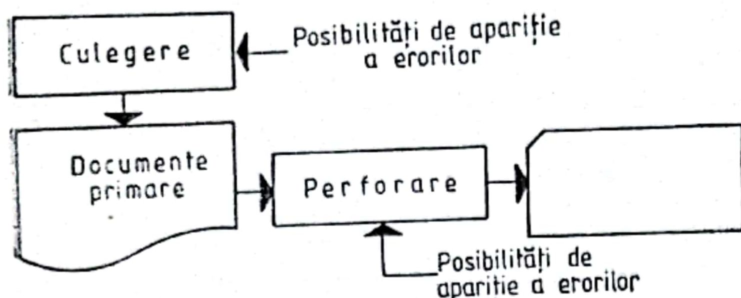


Fig. 3.8.

natură numerică și valori mai mari decît zero (vezi figura 3.8); prin această precizare se asigură un control privind corecta preluare a datelor din documentele primare.

Succesiunea operațiilor va fi următoarea:

```
Op0    început
       0 → V
Op1 — dacă CANTITATE nenumeric
                  atunci
                  1.1. scrie 'CANTITATE NENUMERICA'
                  1.2. 1 → V
                  altfel
                  1.3. dacă   CANTITATE < 1'
                              atunci
                  1.4.              scrie 'CANTITATE < 1'
                  1.5.              1 → V
                                    altfel (*)
                        sf—dacă
       sf—dacă
Op2 — dacă PRET  nenumeric
                atunci
                2.1. scrie 'PRET NENUMERIC'
                2.2. 1 → V
                altfel
                2.3. dacă   PRET < 1
                            atunci
                2.4.        scrie 'PRET < 1'
                2.5.        1 → V
                         altfel
       sf—dacă           sf—dacă
Op3 — dacă V = 1
                atunci
           3.1. scrie 'OPERATII LIPSITE DE SENS'
                altfel
                3.2. CANTITATE * PRET → VALOARE
                3.3. VALOARE + TOTAL—VALOARE → TOTAL—VALOARE
                3.4. 1 + NUMAR—CURENT      → NUMAR—CURENT
                3.5. COD—MATERIAL          → COD—MATERIAL—L
                3.6. DENUMIRE—MATERIAL     → DENUMIRE—MATERIAL—L
                3.7. CANTITATE             → CANTITATE—L
                3.8. PRET                  → PRET—L
                3.9. VALOARE               → VALOARE—L
                3.10. NUMAR—CURENT         → NUMAR—CURENT—L
       sf—dacă
Op.4   sfîrșit
```

Pentru a da posibilitatea unei urmăriri mai ușoare a exemplului, tuturor operațiilor li s-a acordat un număr de secvență.

3.26 După cum se poate observa s-a realizat o delimitare mai întîi în operații de structură generală (Op1, Op2, Op3) și după aceea în operații elementare:

variabila V permite ca prin valorile 0 și 1 să se detecteze dacă a existat sau nu minim o eroare în urma efectuării operațiilor de control asupra datelor (1, a existat minim o eroare; 0 nu existat eroare).

3.27 *Structura repetitivă* indică repetarea unei operații sau a unui grup de operații cît timp o anumită condiție este îndeplinită:

cît timp condiție		
execută		*repetă* procedură
. . .	sau	
sfîrșit		*pînă-cînd* condiție

Structura respectivă poate fi condiționată anterior (întîi se testează condiția și numai dacă răspunsul logic este „fals" se execută operațiile, care definesc procedura) sau posterior (întîi se execută procedura și după aceea se testează condiție pentru a decide asupra repetării ei).

3.28 Dacă spre exemplu dorim să prelucrăm un set de 10 articole perforate în 10 cartele va trebui să repetăm operațiile de prelucrare la nivel articol de un număr de ori egal cu 10.

3.29 *Variantă fără utilizarea structurii repetitive* (fig. 3.9).

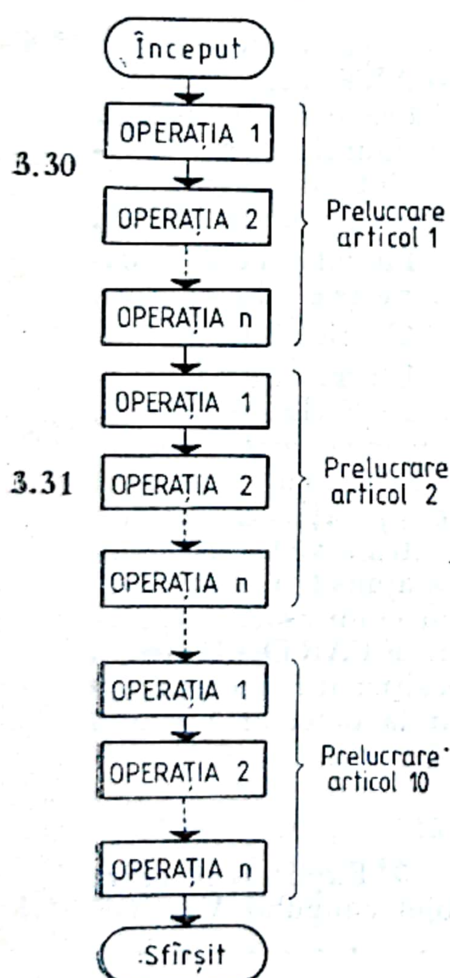


Fig. 3.9.

Folosind structura repetitivă, aceleași operații de prelucrare sînt grupate într-o procedură la care se face apel conform structurii repetitive (fig. 3.10).

3.30 În procesul de descriere a logicii programelor se folosesc cele trei structuri de bază numite și structuri de control, obținîndu-se astfel descrieri ale algoritmilor în conformitate cu conceptele programării structurate. Astfel folosind structuri repetitive se poate face apel la o procedură care la rîndul ei conține alte structuri repetitive, structuri alternative și structuri secvențiale (numite și liniare).

3.31 *2.* Să se descrie grafic structura algoritmului de prelucrare automată a datelor dintr-un

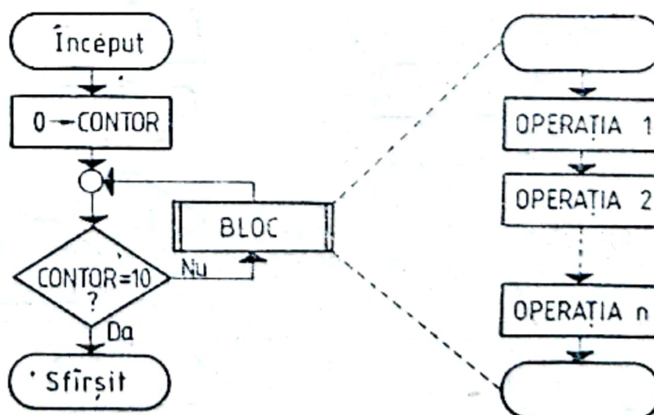


Fig. 3.10.

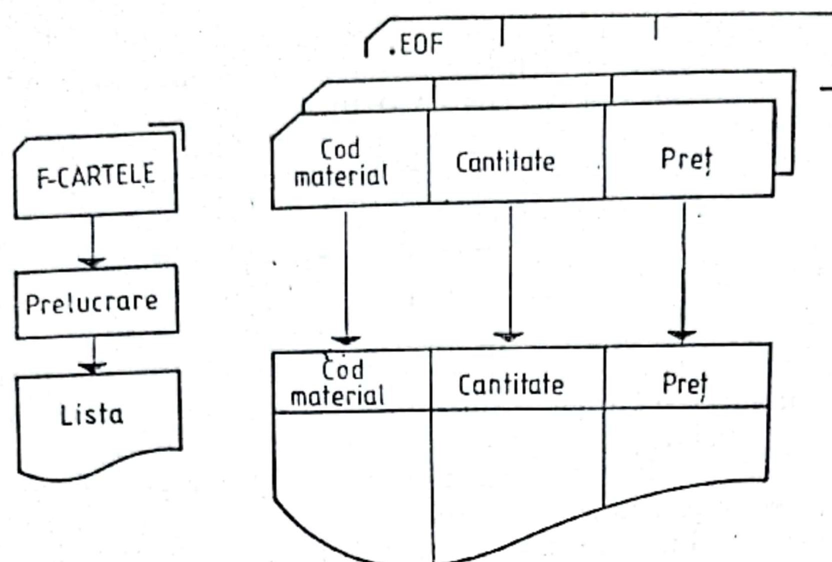
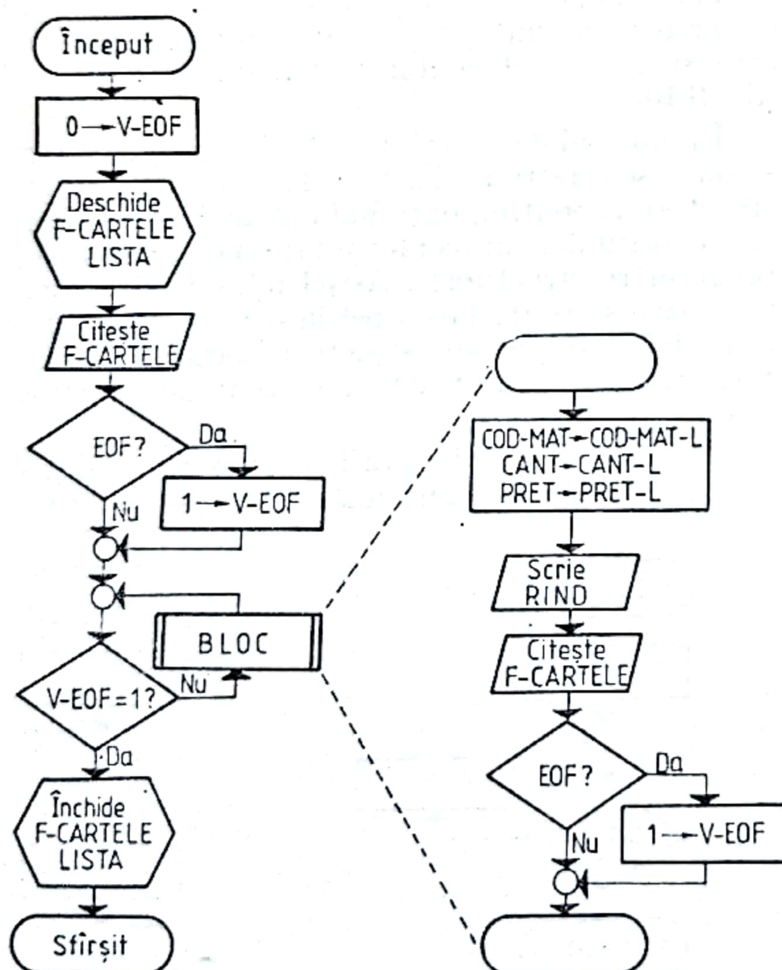


Fig. 3.11.



3.32

Fig. 3.12.

fişier pe cartele numit F-CARTELE care să aibă ca obiectiv listarea conţinutului fiecărei cartele într-un fişier la imprimantă numit LISTA.

Fluxul general al datelor este cel din fig. 3.11.

R. fig. 3.12

Ultima cartelă este EOF (END OF FILE). Comanda *citeşte* va fi însoţită aşa cum am văzut de operaţia de testare pentru a vedea dacă nu s-a ajuns la marca EOF. Un cîmp asociat fişierului F-CARTELE va fi poziţionat la început la *0* iar la detectarea mărcii EOF la *1*.

Î. Explicaţi pe scurt rolul cîmpului V—EOF

. . . .
. . . .
. . . .

R. Cîmpul V—EOF marchează prin valoarea 1 detectarea sfîrşitului de fişier ceea ce permite ieşirea din structura repetitivă.

3.33 *Î.* Cum motivaţi apariţia de două ori a comenzii citeşte F—CARTELE?

. . . .

. . . .

. . . .

R. Întrucît prelucrarea se realizează folosind structura repetitivă, nu se mai revine la prima comandă *citeşte*, care testează doar existenţa datelor în fişier şi crează premisele folosirii structurii repetitive.

3.34 În reprezentarea grafică a algoritmului s-au folosit anumite simboluri grafice. Semnificaţia acestora este redată în tabelele 3.1.a şi 3.1.b. (p.56)

3.35 *Î.* Se cere descrierea grafică a algoritmului care realizează însumarea primelor 100 numere şi afişarea rezultatului la imprimantă.

R. (fig. 3.13).

3.36 Prin reprezentarea grafică a algoritmilor de prelucrare automată a datelor folosind simbolurile descrise în tabelul 3.1.b. se obţine ceea ce literatura de specialitate numeşte *schemă logică de program*. Cînd se descrie fluxul general al datelor se folosesc anumite simboluri (tabelul 3.1.a.) şi se obţine *schema logică de sistem*.

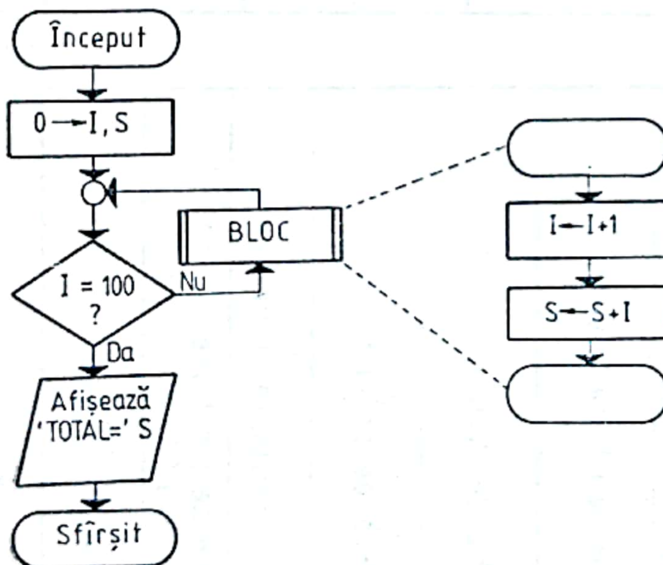


Fig. 3.13.

Exemplul (fig. 3.14).

3.37 Descrierea de detaliu a procesului de prelucrare a datelor (a algoritmului) se realizează cu ajutorul *schemei logice de program*.

Spre exemplu, algoritmul de prelucrare a datelor din FIŞIER—BANDA în vederea obţinerii la imprimantă a situaţiei centralizatoare a materialelor intrate (fig. 3.15.) în cursul unei perioade de gestiune este prezentat grafic în schema logică de programare din figurile 3.16a, 3.16b, 3.16c (pag. 58).

3.38 *Î.* Care sînt structurile de bază utilizate în conceperea programelor

. . . .

. . . .

. . . .

Tabelul 3.1.

Simbol	Semnificație
	Operație de calcul și atribuire
	Operații de deschidere/închidere fișiere
	Pune în evidență începutul și sfîrșitul unei scheme logice
	Operație de intrare/ieșire
	Operație de decizie
	Procedură de prelucrare
	Sensul de derulare a prelucrării
	Punctele de joncțiune din cadrul schemei logice
	Conector de pagină

a.

Simbol	Semnificație
	Cartelă perforată
	Fișier pe cartele
	Document-situație
	Bandă de hîrtie
	Bandă magnetică
	Disc magnetic
	Proces de calcul (unitate de prelucrare; procedură; program)
	Operație de sortare
	Operație de fuzionare, interclasare
	Introducere manuală a datelor
	Conector de pagină

b.

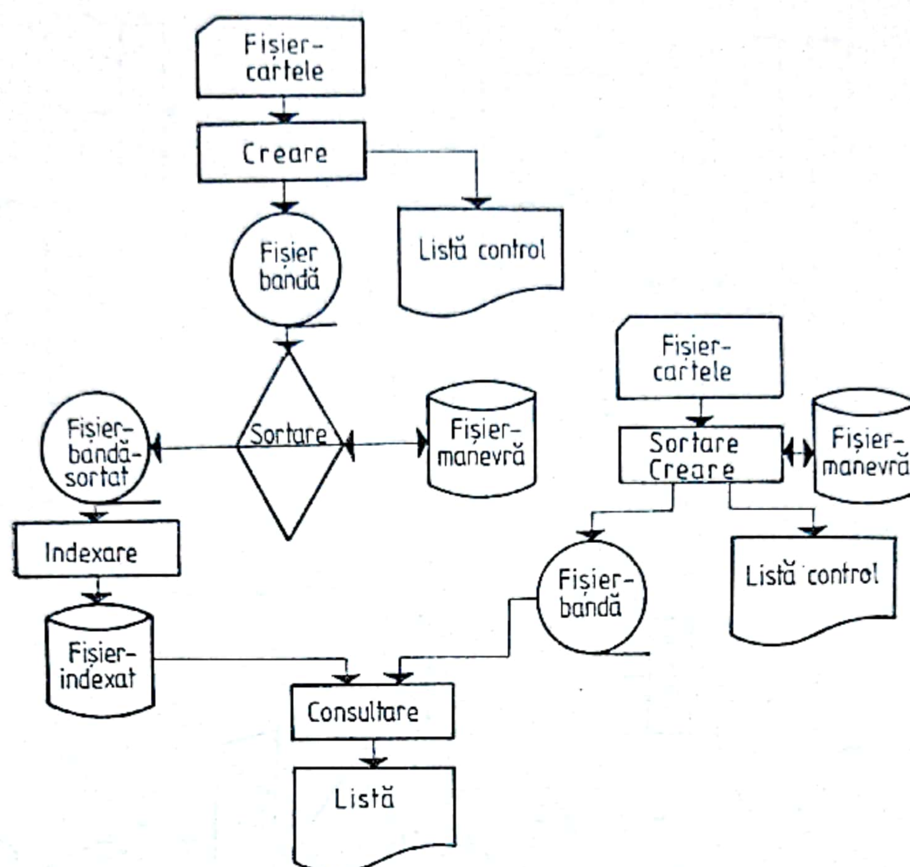


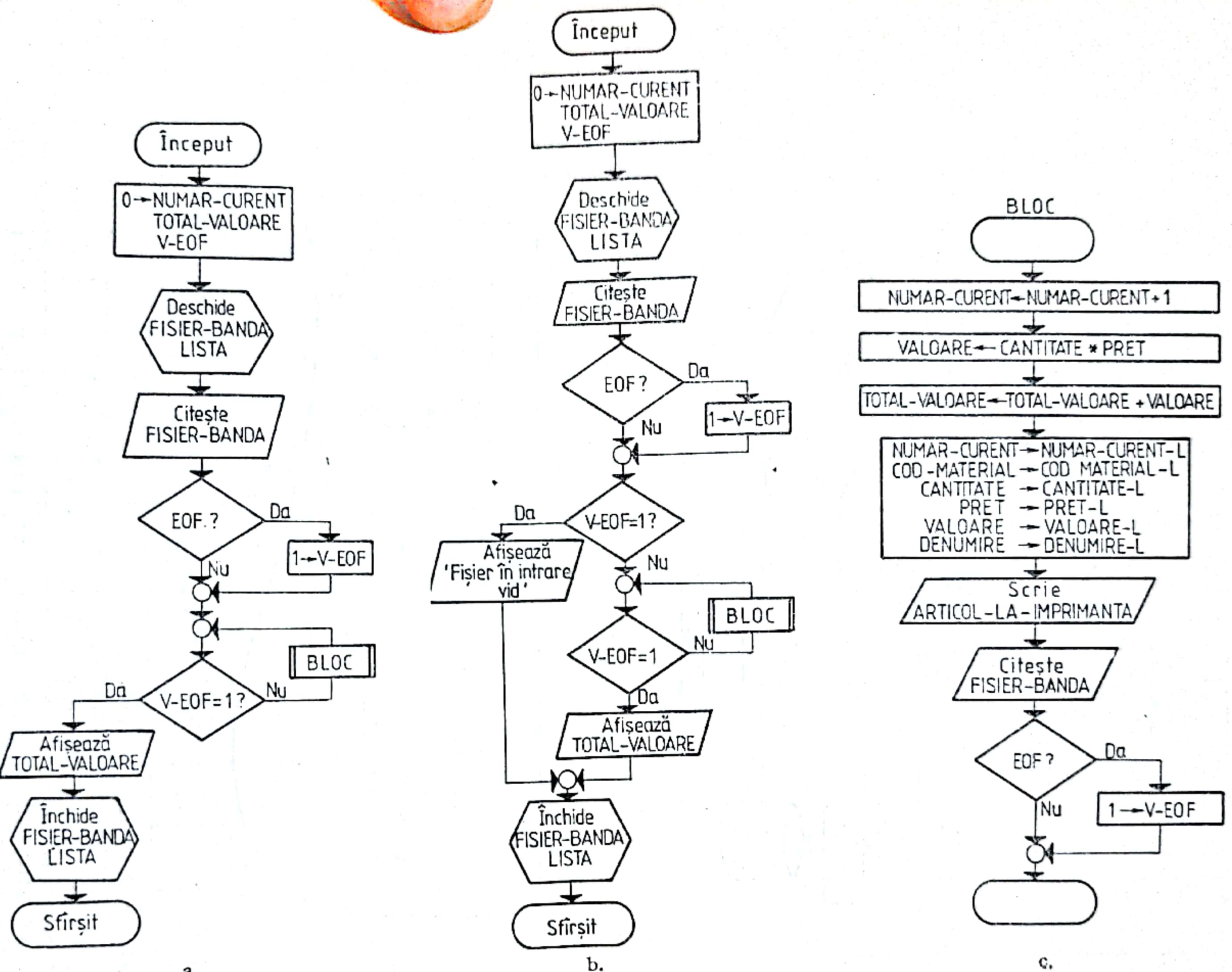
Fig. 3.14.

1	TABLA GALVANIZATA	50031	11.50	15.00	172.50
2	TABLA GALVANIZATA	50031	11.50	20 00	230.00
3	TEAVA 4 MM	50033	12.50	30.00	375.00
4 .	SIRMA DIN OTEL ALIAT	50043	51.20	9.00	460.80
5	TABLA NEAGRA	50034	5:70	150.00	855.00
	TOTAL VALOARE				8.970.20

Fig. 3.15.

R. — structura secvenţială
— structura alternativă
— structura repetitivă.

3.39 În condiţiile în care se cere gruparea valorică pe fel de material (fig. 3.17) se obţine schema logică de programare din fig. 3.18.


Început
0→NUMAR-CURENT
TOTAL-VALOARE
V-EOF
Deschide
FISIER-BANDA
LISTA
Citește
FISIER-BANDA
EOF ?
Da
1→V-EOF
Nu
BLOC
V-EOF=1?
Da
Nu
Afișează
TOTAL-VALOARE
Închide
FISIER-BANDA
LISTA
Sfîrșit
Afișează
'Fișier în intrare vid'
V-EOF=1
NUMAR-CURENT←NUMAR-CURENT+1
VALOARE←CANTITATE * PRET
TOTAL-VALOARE←TOTAL-VALOARE + VALOARE
NUMAR-CURENT→NUMAR-CURENT-L
COD-MATERIAL→COD MATERIAL-L
CANTITATE→CANTITATE-L
PRET→PRET-L
VALOARE→VALOARE-L
DENUMIRE→DENUMIRE-L
Scrie
ARTICOL-LA-IMPRIMANTA
a.
b.
c.
Fig. 3.16.

Antet		CENTRALIZATORUL IESIRILOR DE MATERIALE						R1
		************************************						R2
	= = = =	= = = = = = = =	= = = =	= =	= = = =	= = = =	= = = =	R3
	NRC	DENUMIRE MATERIAL	COD-MAT	UM	CANT	PRET	VALOARE	R4
	= = = =	= = = = = = = =	= = = =	= =	= = = =	= = = =	= = = =	R3'
Grupa de control 1	1	TABLA GALVANIZATA	500 31	KG	15.50	11.50	172.50	
	2	TABLA GALVANIZATA	500 31	KG	20.00	11.50	230.00	
	3	TABLA GALVANIZATA	500 31	KG	30.00	11.50	345.00	
		TOTAL FEL MATERIAL					747.50	
Grupa de control 2	4	SIRMA DIN OTEL A	600.11	KG	3.00	50.00	150.00	
	5	SIRMA DIN OTEL A	600.11	KG	5.00	50.00	250.00	
		TOTAL FEL MATERIAL					400.00	
	- - - -	- - - - - - - -	- - - -	- -	- - - -	- - - -	- - - -	
		TOTAL FEL MATERIAL					260.00	
		TOTAL GENERAL					2.460.00	

Fig. 3.17.

Pentru a da posibilitatea comparării valorii curente a indicatorului de grupare (codul materialului din articol citit) cu valoarea precedentă (codul materialului citit anterior) în vederea detectării cazului în care s-a citit un articol ce se referă la un alt fel de material (cu alt cod) se definește un cîmp în memoria de lucru (numit de noi ZONA—COD—MAT) în care se transferă prima valoare a cîmpului COD—MAT din fiecare grupă de control. *Numim grupă de control ansamblul articolelor care cu aceeași valoare pentru indicatorul de grupare.* Sînt cazuri cînd o grupă de control conține un singur articol în fișierul de intrare. La terminarea unei grupe de control trebuie realizată editarea totalului. Se poate constata că s-a terminat o grupă de control după ce s-a citit un articol din grupa de control următoare, caz în care la realizarea testului

COD—MAT = ZONA—COD—MAT? se răspunde „NU".

Procesul de prelucrare se descompune în trei module*) și anume:

— modulul principal (director) (fig. 3.18.a.);
— modulul fel—material (fig. 3.18.b);
— modulul operațional (fig. 3.18.)c., de la pag. 61.

Pentru a ieși din structura repetitivă care activează modulul operațional s-a definit un cîmp VARIABILA—SFM care ia valoarea 1 cînd s-a citit un articol care face parte dintr-o nouă grupă de control.

*) În accepțiunea autorilor termenul de modul este echivalent celui de procedură.

Modulul operațional cuprinde trei categorii de operații:
— operații de prelucrare a articolului citit;
— operații de citire a unui nou articol;
— operații de decizie pentru a vedea dacă s-a trecut la o nouă grupă de control.

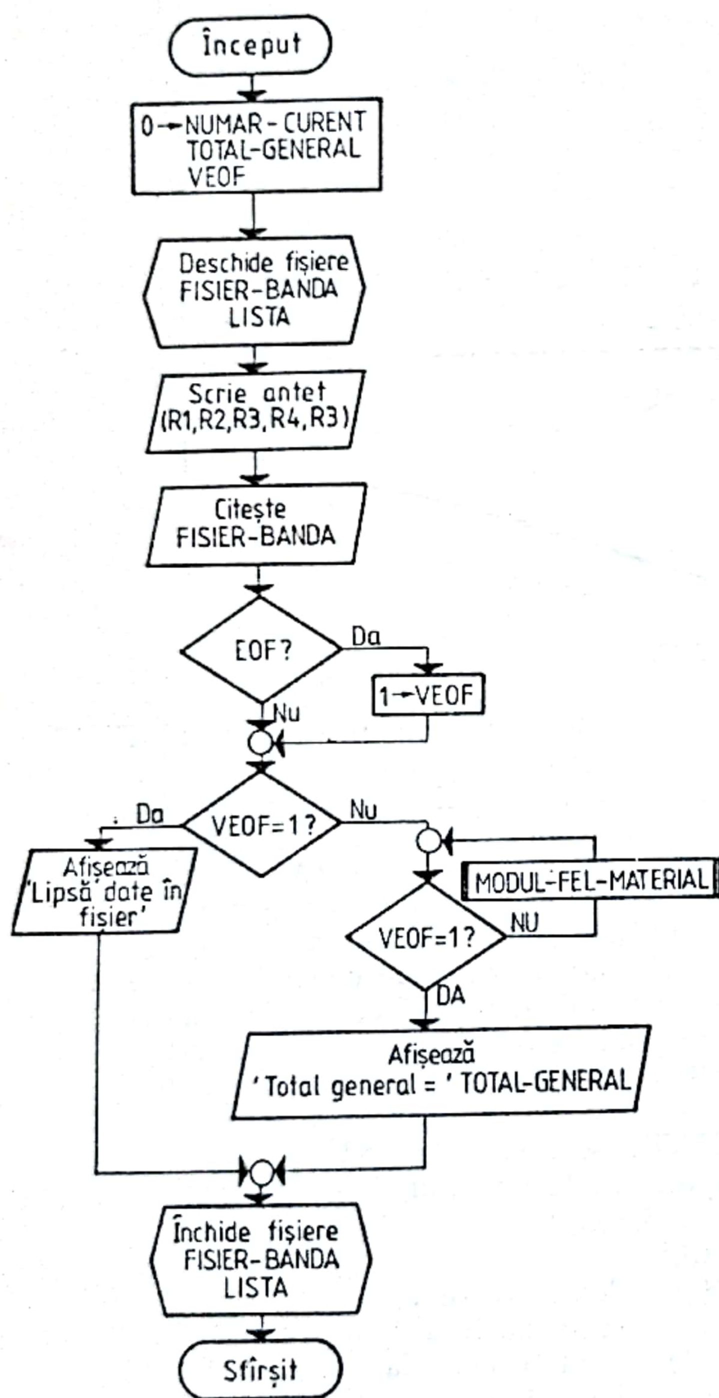


Fig. 3.18. a

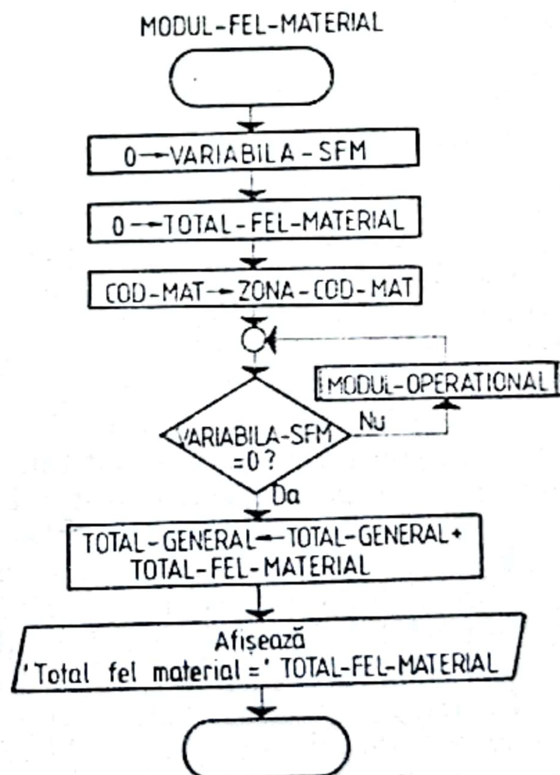


Fig. 3.18. b

→

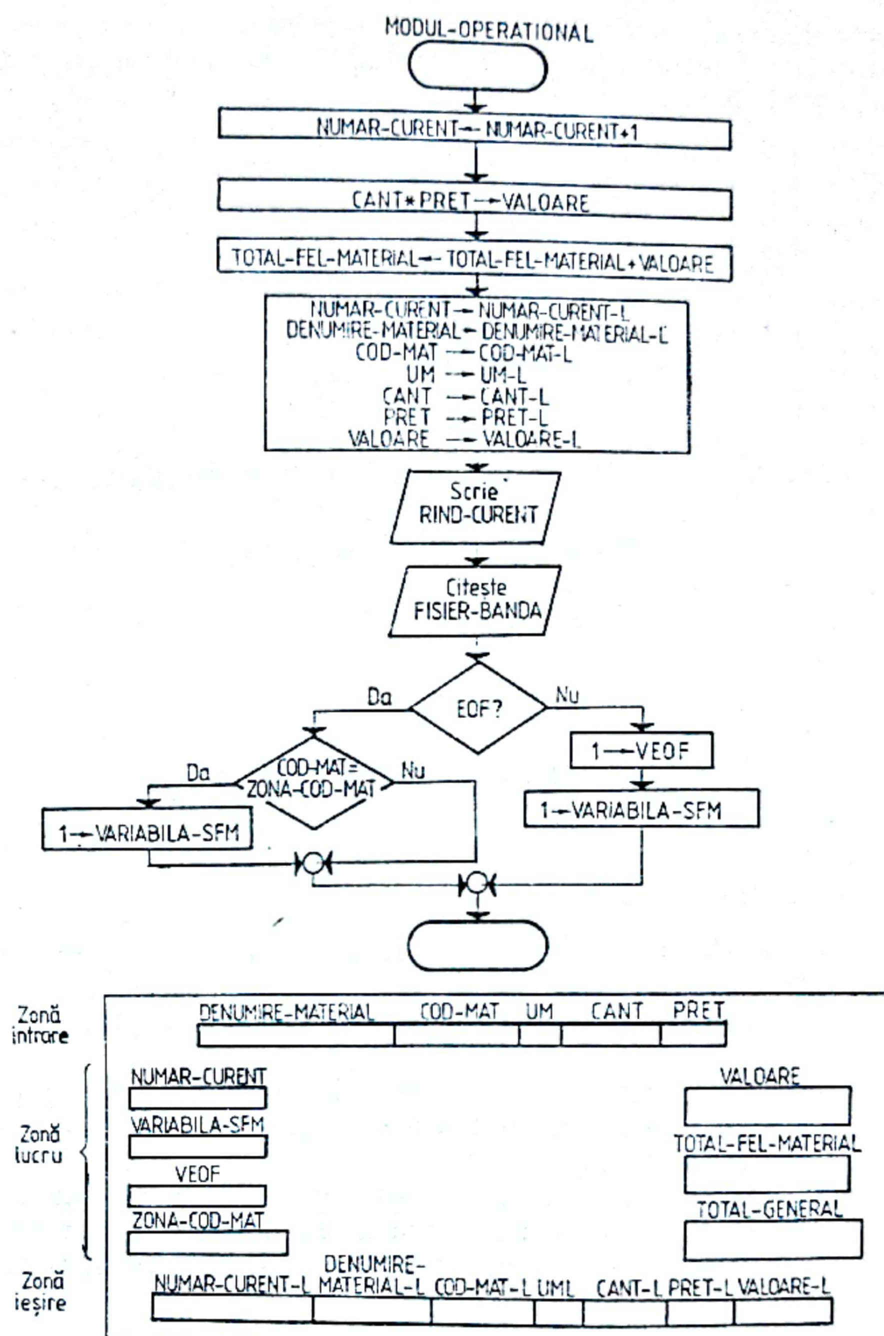
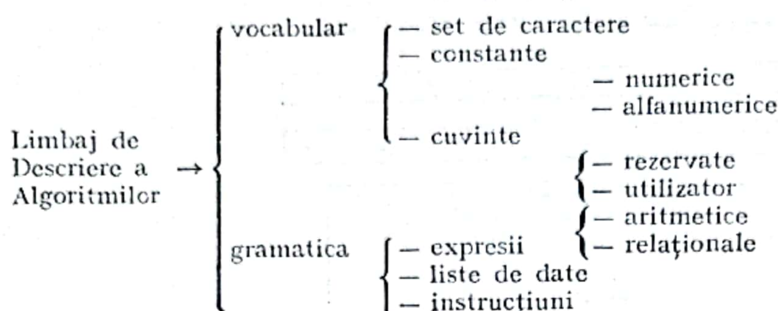


Fig. 3.18. c

3.40 În reprezentarea algoritmilor se poate folosi una din tehnicile:
— schema logică;
— limbajul de tip pseudocod.

3.41 Limbajul de tip pseudocod recomandat în lucrare a fost conceput în ideea apropierii de particularitățile programării folosind limbajul COBOL. Avantajul folosirii limbajelor de tip pseudocod constă în faptul că acestea deschid perspectiva realizării unor compilatoare care să facă trecerea automată a programelor din limbaj de tip pseudocod în limbaj evoluat.

3.42 Structura generală a limbajului de tip pseudocod este asemănătoare celei definite pentru limbajul LDA (Limbaj de Descriere a Algoritmilor) în cadrul I.C.I de St. Niculescu [12].



Setul de caractere este același cu cel folosit în limbajul COBOL (vezi lecția 4).

3.43 *Constantele numerice* sînt succesiuni de caractere numerice.

Exemple:

343
3555,34
—35,43
—35

Constantele alfanumerice sînt șiruri de caractere alfanumerice care apar între apostrofuri.

Exemplu:

'TOTAL FEL MATERIAL = '

3.44 *Cuvintele rezervate* reprezintă succesiuni de caractere (maximum 30) care în descrierea algoritmului apar cu minuscule cursive. Ele au semnificație prestabilită și reprezintă codul operațiilor sau opțiunilor din cadrul acestora.

Cuvintele utilizator reprezintă succesiuni de maximum 30 caractere; apar în descrierea algoritmului cu majuscule și sînt numite așa pentru că sînt alese de programator.

3.45 *Expresiile aritmetice* sînt cele cunoscute din prelucrarea manuală a datelor numai că ele capătă o scriere liniară. În acest sens pentru operația de împărțire se utilizează simbolul '/' iar pentru ridicare la putere caracterele '**'.

Exemple:

A = B + C × D + E ** 2 + 1 000
F = B * C / D

3.46 *Expresiile relaționale* se obțin cu ajutorul operatorilor relaționali <, >, =, NOT (negația), SI (conjuncția), SAU (disjuncția).

Exemple:

CANTITATE > 1 000
CANTITATE—1 < CANTITATE—2
A ** 2 NOT = D ** 2
CANTITATE > 0 sau < 10 000

3.47 *Instrucțiunile* desemnează operațiile de prelucrare precizate prin cuvintele rezervate:

— *început*, pentru a marca începutul descrierii unui algoritm;

— *cîmpuri*, pentru a desemna cîmpurile intermediare folosite în prelucrare (utilizarea sa este opțională);

— *citește*, pentru a desemna operația de introducere a articolelor de pe suporturi tehnice în memoria centrală;

— *scrie*, pentru a desemna operația de scriere a articolelor pe suport tehnic;

— *calculează*, pentru a desemna operațiile de calcul;

— *dacă*, pentru a desemna operațiile de decizie (el permite deci descrierea structurii alternative);

— *repetă*, pentru a desemna o structură repetitivă într-un algoritm;

— *atribuie* → sau :=, pentru a desemna transferul unei date dintr-o zonă de memorie în alta (dintr-un cîmp în altul);

— *procedură*, pentru a desemna o procedură ce va fi descrisă în altă parte a algoritmului;

— *sfîrșit*, pentru a desemna sfîrșitul logic în prelucrare.

3.48 Descrierea în limbajul de tip pseudocod LDA a algoritmului reprezentat în figura 3.16 este următoarea:

```
început  (POPESCU, ALG10, 3458)
         0 → NUMĂR–CURENT
             TOTAL–VALOARE
             V–EOF
         citește      FIȘIER–BANDA
                          dacă–marca EOF
                                  1 → V–EOF.
         dacă         V–EOF = 1
                      atunci
                          scrie 'FIȘIER ÎN INTRARE VID'
                      altfel
                          repetă # procedură PROCO1
                                   pînă–cînd V–EOF = 1
                          scrie    'TOTAL GENERAL = ' TOTAL–VALOARE
         sf–dacă
      sfîrșit
```

```
procedură PROCO1;
   început–procedură
         1 + NUMĂR–CURENT → NUMĂR–CURENT
         CANTITATE * PREȚ → VALOARE
         VALOARE + TOTAL–VALOARE → TOTAL–VALOARE
         NUMĂR–CURENT      → NUMĂR–CURENT     din RÎND–LISTA
         COD–MATERIAL      → COD–MATERIAL     din RÎND–LISTA
         CANTITATE         → CANTITATE        din RÎND–LISTA
         PREȚ              → PREȚ             din RÎND–LISTA
```

```
        VALOARE          → VALOARE      din RÎND—LISTA
        DENUMIRE         → DENUMIRE     din RÎND—LISTA
    scrie RÎND—LISTA
    citește FIȘIER—BANDA
                           dacă—marca—EOF
                              1→ V—EOF.
sfîrșit—procedură PROCO1
```

3.49 Dacă schema logică de program constituie o formă de reprezentare sugestivă și deja asimilată în practica descrierii algoritmilor, limbajul de tip pseudocod oferă avantaje atît pe linia descrierii în etape și pe nivele ierarhice de detaliere a programelor cît mai ales pe linia autodocumentării acestora. Totodată, faptul că limbajul LDA a fost orientat spre operații similare celor realizate în limbajul COBOL, oferă perspectiva elaborării unor programe de tipul compilatoarelor capabile să automatizeze în mare parte munca propriu-zisă de programare.

3.50 *Î.* Descrieți schema logică de programare pentru obținerea situației centralizatoare a intrărilor de materiale în gestiunea 05 cunoscînd următoarele date:

— schema logică de sistem (fig. 3.19);
— structura datelor în intrare și respectiv în ieșire (fig. 3.20).

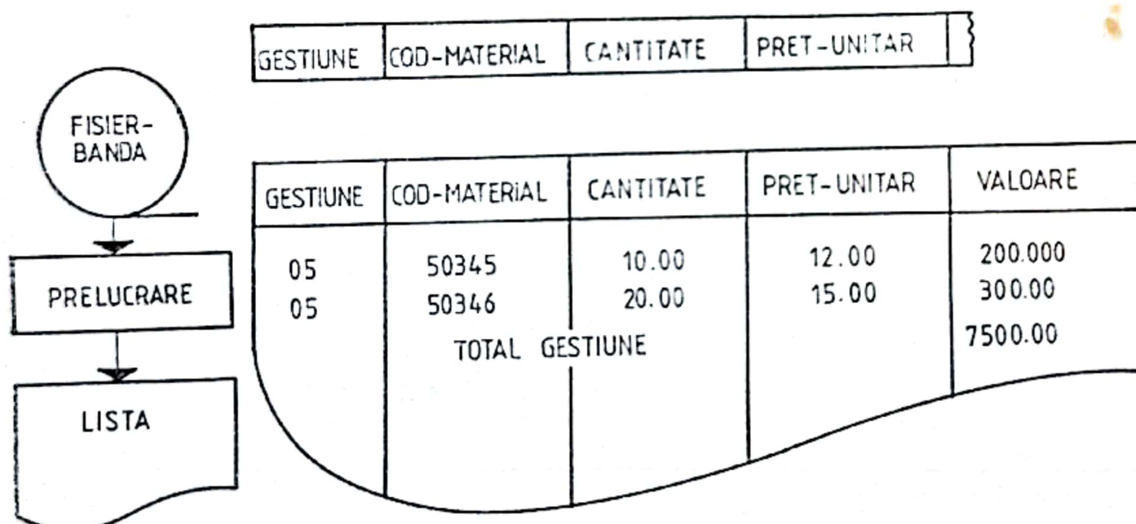


Fig. 3.19. Fig. 3.20.

Fișierul FIȘIER—BANDA cuprinde date privind toate gestiunile întreprinderii.

R. (fig. 3.21)

Răspunsul este asemănător cu cel din fig. 3.21.

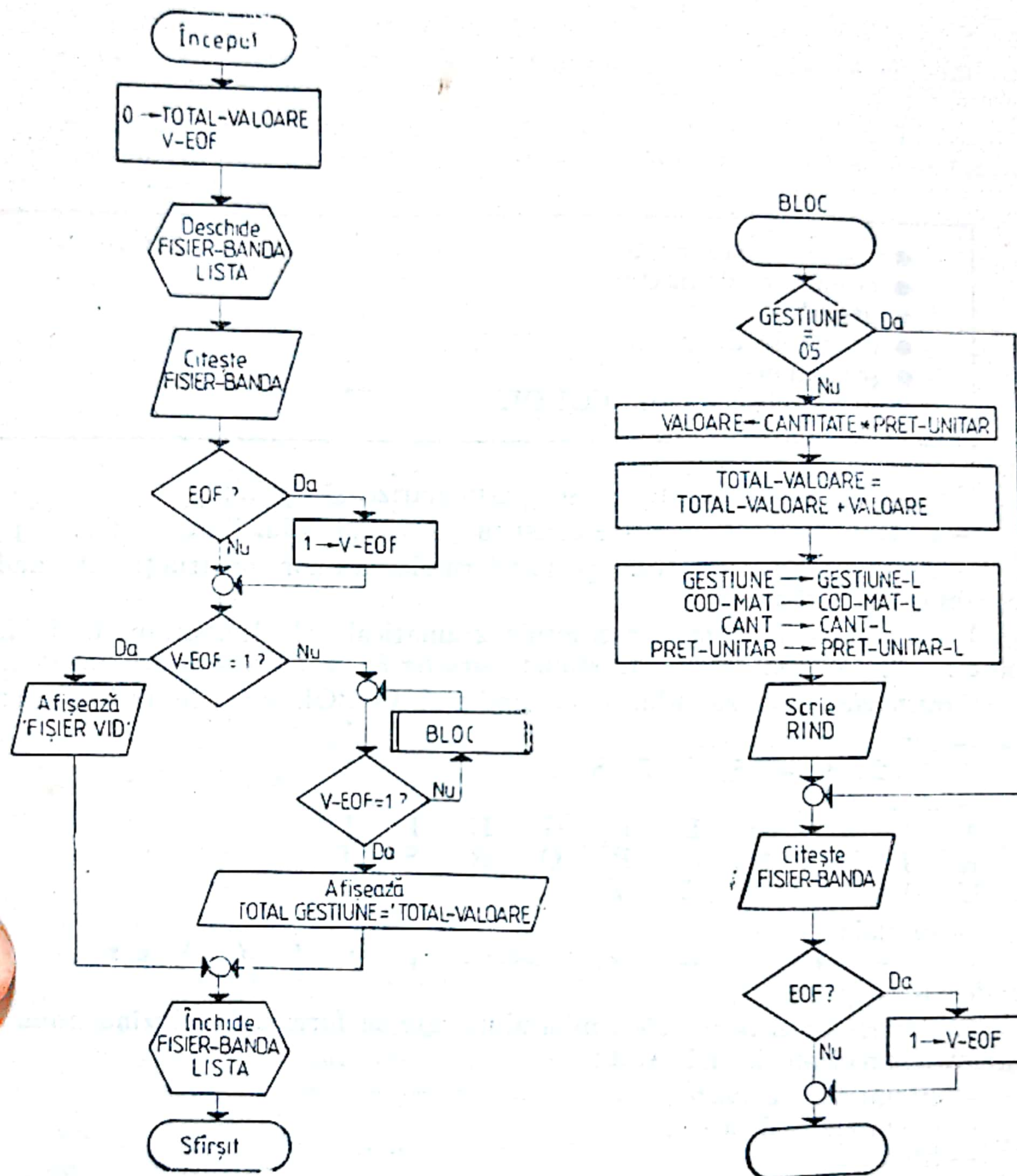


Fig. 3.21.

DA → Lecţia 4
NU ↰ 3.9.

LECŢIA a 4-a ELEMENTE CONSTITUTIVE ALE LIMBAJULUI COBOL. FORMATE GENERALE COBOL

- cuvintele rezervate
- cuvintele utilizator
- literalele
- numerele de nivel
- şabloanele
- formatele generale COBOL

4.1 Un limbaj de programare se particularizează prin:

— mulţimea cuvintelor care constituie vocabularului limbajului;

— regulile gramaticale care permit formularea unor construcţii, utilizînd vocabularul limbajului.

Înainte de a analiza construcţiile gramaticale ale limbajului COBOL prezentăm componentele acestuia: *caracterele* şi *elementele constitutive.*

4.2 *Caracterele* de bază admise de limbajul COBOL sînt de trei tipuri:

— numerice

0 1 2 3 4 5 6 7 8 9

— alfabetice

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z				

— speciale

+ — * / = < > . , ; ' ( ) ƀ $

unde: ƀ = spaţiu

4.3 *Elementele constitutive* ale limbajului, care se formează utilizînd numai caractele precizate la 4.2, sînt:

— cuvintele rezervate;

— cuvintele utilizator;

— literalele;

— numerele de nivel;

— şabloanele.

4.4 *Cuvintele rezervate* sînt succesiuni de caractere care au semnificaţie prestabilită în cadrul limbajului; utilizatorul nu poate modifica şi nici da altă utilizare acestor cuvinte.

Exemple de cuvinte rezervate:

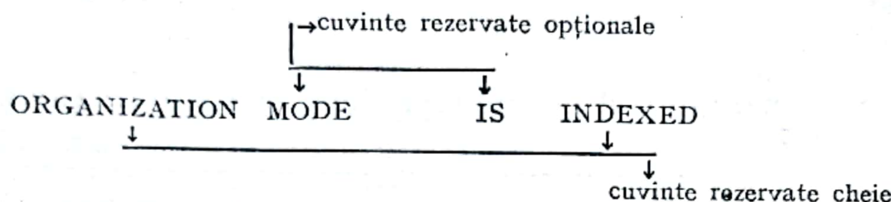
READ — citeşte
WRITE — scrie
MOVE — transferă

Cuvintele rezervate sînt de două tipuri:

— *cuvinte rezervate cheie* — a căror prezență în programul elaborat de utilizator este obligatorie;

— *cuvinte rezervate opționale* — care pot lipsi din program fără a influența calitatea acestuia; ele se folosesc pentru a da un sens mai clar construcției în care se utilizează.

Exemplu:



Utilizatorul va ști de ce tip este un cuvînt rezervat, prin consultarea formatelor generale COBOL, care vor fi comentate în paragrafele 4.13—4.23 și prezentate în anexa 1.

4.5 *Î.* Cuvintele rezervate sînt . . . . și se pot grupa în cuvinte . . . . și . . . .

R. — prestabilite
— cheie
— opționale

4.6 *Cuvintele utilizator* sînt formate de programator (utilizator) pentru a denumi fișiere, articole, cîmpuri de date, paragrafe, secțiuni etc.

Unui fișier privind stocurile de materiale i se va da, spre exemplu, numele FIȘIER—STOCURI.

Exemple de cuvinte utilizator:

— FIȘIER—CARTELE, pentru a desemna numele unui fișier pe cartele;

— FIȘIER—STOCURI, pentru a desemna numele unui fișier de stocuri de valori materiale:

— VALOARE, pentru a desemna numele unui cîmp;

— P123, pentru a desemna numele unui paragraf sau al unei secțiuni;

— 153, pentru a desemna numele unui paragraf sau al unei secțiuni, etc.

La formularea cuvintelor utilizator trebuie respectate următoarele reguli:

— să difere de cuvintele rezervate;

— să cuprindă numai litere, cifre și liniuța de unire;

— liniuța de unire nu poate fi utilizată la începutul sau sfîrșitul cuvîntului;

— în interiorul unui cuvînt nu se poate folosi spațiul;

— un cuvînt poate avea maximum 30 caractere;

— să fie semnificative pentru a permite obținerea unor programe accesibile și ușor de întreținut;

— cuvintele nume de paragraf sau de secţiune pot fi formate şi numai din cifre *.

4.7 *Î.* Indicaţi care cuvinte utilizator din cele ce urmează, sînt corecte?
1. PREŢ—UNITAR
2. COD MATERIAL
3. VALOARE
4. UNITATE—DE—MĂSURĂ
5. 12345
6. READ
7. CANT

R. 1, 3, 4, 5 (numai în cazul în care reprezintă un nume de paragraf sau de secţiune) şi 7

4.8 *Literalele* sînt şiruri de caractere, compuse de programator, ale căror valori sînt date prin înseşi caracterele din care se compun. În general, în limbajele de programare literalele au semnificaţie de constante.

Există trei tipuri de literale:
— literale numerice;
— literale nenumerice;
— literale speciale (constante figurative).

Literalele numerice sînt şiruri de maximum 18 caractere compuse din cifre (0—9) semnele + şi — şi semnul pentru marcarea zecimalelor (punctul zecimal).

Semnele + sau — se plasează în extremitatea stîngă a literalului (cînd lipseşte se consideră semnul +). Punctul zecimal se plasează în interiorul sau la începutul şirului de cifre şi poate să apară cel mult o dată.

4.9 *Î.* Precizaţi care din următoarele elemente pot fi literale numerice;
1979
+32.25
03.12.78
±15

R. 1979
+32.25

4.10 Literalele numerice mai pot fi exprimate şi sub forma:
[±] mantisă E [±] exponent
unde:
— *mantisa* reprezintă o succesiune de maxim 16 cifre (care trebuie să includă şi marca zecimală);
— *exponent*, un număr întreg format din una sau două cifre.

Exemplu:
125.06E—2

---
* Diviziunea PROCEDURE a unui program se poate compune din secţiuni (definite de programator) şi paragrafe.

4.11 *Literalele nenumerice* sînt succesiuni de maximum 120 caractere delimitate cu apostrofuri.

Exemplu:

MOVE 'BALANŢA CONTURILOR SINTETICE' TO CÎMP

→literal nenumeric

În urma acestei instrucţiuni zona CÎMP ia valoarea:

| |B|A|L|A|N|Ţ|A| |C|O|N|T|U|R|I|L|O|R| |S|I|N|T|E|T|I|C|E|

Apostroful nu poate figura printre caracterele literalului nenumeric.

4.12 *Literalele speciale* sînt cunoscute şi sub numele de constante figurative; valoarea constantei este precizată cu ajutorul cuvintelor rezervate.

Principalele constante figurative sînt:

— ZERO (ZEROS, ZEROES), unul sau mai multe caractere zero;
— SPACE (SPACES), unul sau mai multe caractere spaţiu;
— HIGH—VALUE (HIGH—VALUES), unul sau mai multe caractere a căror reprezentare în hexazecimal este FF;
— LOW—VALUE (LOW—VALUES), unul sau mai multe caractere a căror reprezentare în hexazecimal este OO;
— QUOTE (QUOTES) unul sau mai multe caractere apostrof;
— ALL 'c' unul sau mai multe caractere desemnate prin c

Exemplu:

MOVE	ZERO	TO	CÎMP
↓	↓	↓	
transferă	constantă figurativă	în	

Lungimea constantei figurative este egală cu lungimea cîmpului căruia îi este asociată printr-o instrucţiune COBOL.

Fie cîmpul ZONA de natură alfanumerică, lungime 5 caractere şi valoare COBOL:

ZONA

|C|O|B|O|L|

În instrucţiunea:

MOVE	SPACES	TO	ZONA
(transferă)		(în)	

constanta figurativă SPACES are lungimea 5 iar efectul instrucţiunii este:

ZONA

|ƀ|ƀ|ƀ|ƀ|ƀ|

Formele de singular şi plural ale constantelor figurative sînt echivalente.

4.13 *Numerele de nivel* sînt grupuri de caractere numerice care indică ierarhia datelor.

Dacă macheta articolului asociat unui fişier pe cartele este asemănătoare celei din figura 4.1. ierarhia datelor se prezintă ca în figura 4.2.

Pentru a descrie ierarhia datelor în cadrul articolului se folosesc numerele de nivel asociate fiecărui nume de dată (cîmp).

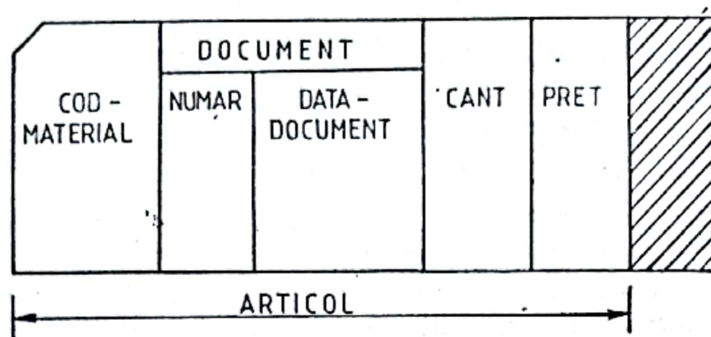


Fig. 4.1.

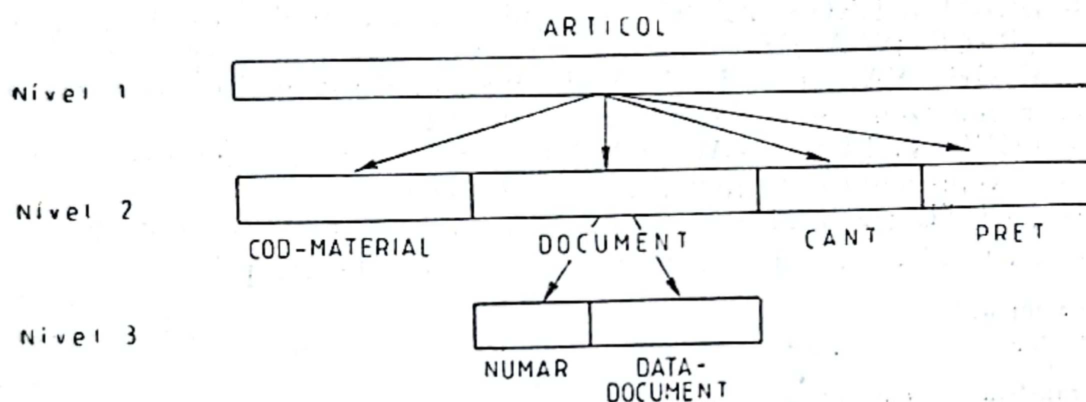


Fig. 4.2.

Exemplu:

```
01      ARTICOL.
        02    COD—MATERIAL ..........
        02    DOCUMENT.
              03  NUMĂR ..........
              03  DATA—DOCUMENT ..........
        02    CANT ............
        02    PRET ............
```

Numărul de nivel 01 se asociază articolelor iar numerele de nivel 02—49, 88 datelor din structura acestora. În descrierea datelor numărul de nivel este urmat de numele datei (cîmpului).

4.14 *Şabloanele* sînt şiruri de caractere; ele precizează natura şi lungimea datelor. După natura lor datele pot fi:

— numerice
  - de calcul
  - de editare

— alfanumerice

— alfabetice

Succesiunea de caractere care apar în mulţimea valorilor posibile defineşte natura datei, iar numărul (maxim) al caracterelor ce o compun lungimea acesteia.

4.15 **Exemple de șabloane:**

— X(10), pentru un cîmp care conține date de natură alfanumerică și lungime 10 caractere;
— 9(5), pentru un cîmp care conține date de natură numerică și lungime 5 caractere;
— A(27), pentru un cîmp care conține date de natură alfabetică și lungime 27 caractere;
— Z(4)9.99 pentru un cîmp numeric de editare de lungime cinci întregi și două zecimale;
— 9(5)V99 pentru un cîmp numeric de calcul de lungime cinci întregi și două zecimale.

Fie cîmpul PRET de natură numerică care poate conține valori de ordinul zecilor de mii (fără virgulă). Descrierea completă este:

02 PRET PIC 9(5).

unde:

PIC este un cuvînt rezervat care precizează că în descrierea datei urmează șablonul ce definește natura și lungimea.

Șablonul poate fi scris și în formă extinsă, astfel: 99999.

În cazul în care cîmpul PRET cuprinde valori de ordinul zecilor de mii la partea întreagă și două zecimale se realizează următoarea descriere:

02 PRET PIC 9(5)V99.

unde:

V indică poziția virgulei zecimale (ea nu ocupă spațiu în memorie).

Din exemplele prezentate rezultă că șablonul precizează natura și lungimea unei date elementare.

În cadrul unui șablon pot apare maxim 30 simboluri cu semnificații speciale.

La formarea șabloanelor trebuie avut în vedere regulile:

— caracterul V indică poziția virgulei zecimale și nu poate fi ultimul simbol al șablonului;

— simbolurile $ și + pot figura o singură dată în șablon și anume la începutul acestuia;

— numărul maxim de caractere 9 ce pot apare în șablon este de 18.

4.16 *Î.* Descrieți cîmpul CANTITATE de natură numerică și lungime patru întregi și trei zecimale.

. . . .

R. 02 CANT PIC 9(4)V999.

4.17 Pentru a descrie un cîmp care conține date de natură alfanumerică se utilizează în șablon caracterul de bază X.

Exemplu:

02 UNITATE—MĂSURĂ PIC XXX.

Un cîmp ce conține date de natură alfabetică se descrie cu caracterul A în șablon.

Exemplu:

02 NUME—PRENUME PIC A(20).

4.18 *Î.* Descrieți cîmpurile DENUMIREA—MATERIALULUI format din 40 caractere alfabetice și STRADA format din 20 caractere alfanumerice (ambele au nivel ierarhic 02) . . . .
. . . .

R.
```
02     DENUMIREA—MATERIALULUI     PIC     A(40).
02     STRADA     PIC     X(20).
```

4.19 Revenind la exemplul de la paragraful 4.13 se ajunge la următoarea descriere completă a articolului:

```
01         ARTICOL.
    02         COD—MATERIAL      PIC        9(7).
    02         DOCUMENT.
        03     NUMĂR             PIC        9(5).
        03     DATA—DOCUMENT     PIC        9(6).
    02         CANT              PIC        9(5)V99.
    02         PRET              PIC        999V99.
    02         PRET              PIC        999V99.
```

În baza acestei descrieri făcută articolului, compilatorul va afecta zonele de memorie necesare pentru stocare.

4.20 Î. Se dă următoarea structură a articolului (fig. 4.3). Realizați descrierea completă utilizînd numerele de nivel, numele de dată, cuvîntul rezervat PIC (acolo unde este cazul) și șabloanele.

. . . .
. . . .

MARCA	NUMELE ȘI PRENUMELE	DATA-OP			RETRIBUȚIA LUNARĂ	
		ZIUA	LUNA	ANUL		
n 4c	a 15c	n 2c	n 2c	n 2c	n 4c	

Fig. 4.3.

R.
```
01         ARTICOL.
           02      MARCA                 PIC       9999.
           02      NUME—PRENUME          PIC       A(15).
           02      DATA—OP.
              03   ZIUA                  PIC       99.
              03   LUNA                  PIC       99.
              03   ANUL                  PIC       99.
           02      RETRIB—LUNARĂ         PIC       9999.
```

4.21 Î. În funcție de natura conținutului lor, datele pot fi:
. . . . caracterul de bază utilizat în șablon fiind. . .;
. . . . . . caracterul de bază utilizat în șablon fiind. . .;

. . . . caracterul de bază utilizat în şablon fiind . . . .

R. — numerice : 9
— alfanumerice : X
— alfabetice : A

4.22 Regulile de combinare a elementelor constitutive ale limbajului pentru a obţine anumite expresii, aşa cum am spus, definesc gramatica limbajului. Pentru fiecare tip de construcţie COBOL va exista cîte o regulă gramaticală.

Limbajul COBOL oferă mai multe posibilităţi de combinare în expresii a elementelor sale constitutive.

Spre exemplu, pentru a realiza operaţia de adunare

TOTAL—VALOARE := TOTAL—VALOARE + VALOARE

sînt două posibilităţi de combinare a cuvintelor COBOL specifice şi anume :

a) ADD VALOARE TO TOTAL—VALOARE
b) ADD VALOARE TOTAL—VALOARE GIVING TOTAL—VALOARE

4.23 Programatorul trebuie să cunoască toate variantele de formulare a unei expresii, fraze, rubrici, instrucţiuni şi, în funcţie de cazul de rezolvat, să aleagă varianta adecvată.

Prin formatele generale COBOL, folosind anumite simboluri speciale şi unele reguli explicative de scriere, se precizează construcţiile gramaticale în formatul general sau în variante posibile.

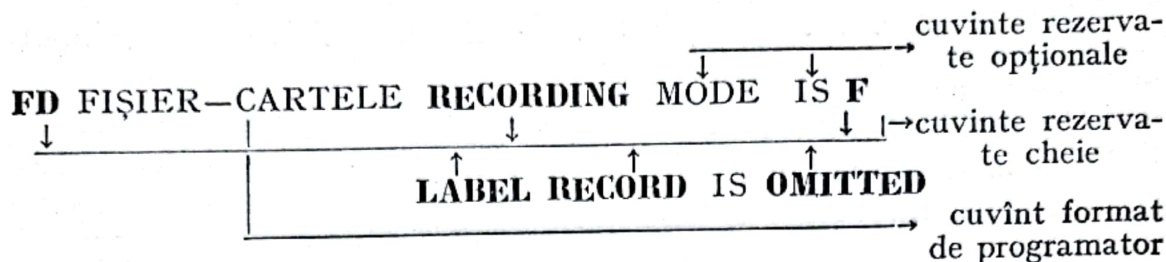
Aceste formate generale de prezentare a diferitelor instrucţiuni, enunţuri şi clauze în una sau mai multe variante, oferă programatorului cadrul comun de aranjare a diferitelor elemente şi simboluri. Ele cuprind elementele constitutive, ordinea lor, plus simbolurile speciale care descriu maniera de a le folosi (metalimbajul). Prin urmare metalimbajul COBOL utilizează simboluri şi reguli pentru alcătuirea construcţiilor ce descriu sintaxa limbajului.

4.24 *Cuvintele scrise cu majuscule* sînt cuvinte rezervate (cele îngroşate-sînt cheie iar cele ne opţionale) şi au o semnificaţie bine definită în limbaj. Acestea pot fi cheie (obligatorii) şi apar îngroşate în formatele generale sau opţionale.

READ FIŞIER—BANDA AT **END MOVE** 1 **TO** VEOF. → cuvinte rezervate cheie

AT → cuvînt rezervat opţional

Cuvintele scrise cu minuscule sînt cuvinte formate de programator în momentul scrierii programului.



Pentru expresia de mai sus, în formatul general, se scrie :

FD nume—fişier **RECORDING** MODE IS **F**
LABEL RECORD IS **OMITTED**.

La scrierea construcţiei FD, *nume—fişier* s-a înlocuit cu numele atribuit de către programator fişierului descris şi anume FIŞIER—CARTELE.

4.25 *Simbolurile metalimbajului* :

a. { } *perechile de acolade*, încadrează unul sau mai multe elem ente dintre care numai unul poate fi utilizat în acel loc al construcţiei.

Exemplu:

```
       ⎧INPUT         {nume-fişier—1}...                  ⎫
       ⎪OUTPUT        {nume-fişier—2}...                  ⎪
OPEN   ⎨     ⎧INPUT—OUTPUT⎫ {nume-fişier—3}...             ⎬...
       ⎪     ⎩   I—O      ⎭                               ⎪
       ⎩                                                  ⎭
```

unde:

- *OPEN*, comanda de deschidere a fişierelor;
- *INPUT*, în intrare (cînd se citesc date din fişier);
- *OUTPUT*, în ieşire (cînd se scriu date în fişier);
- *INPUT—OUTPUT* (I—O) — în intrare/ieşire (cînd se citesc şi se scriu date în fişier).

După caz programatorul va alege :

— dacă vrea să deschidă un fişier în intrare

OPEN INPUT nume—fişier

— dacă vrea să deschidă un fişier în ieşire

OPEN OUTPUT nume—fişier

—*b.* [ ] *parantezele pătrate* încadrează o construcţie COBOL opţională.

Exemplu:

[**AUTHOR.** nume—autor.]

Programatorul poate indica în cadrul programului numele autorului sau îl poate omite; în ambele cazuri efectul este acelaşi.

Sînt cazuri cînd o construcţie, deşi este cuprinsă între paranteze, particularităţile exemplului concret impun utilizarea ei în program. Spre exemplu în formatul de descriere a fişierelor (rubrica FD) apare clauza :

[**BLOCK** CONTAINS întreg **RECORDS**]

Dacă articolele nu sînt blocate clauza nu trebuie folosită ; în caz contrar folosirea clauzei este obligatorie, şi permite să se precizeze numărul de articole dintr-un bloc.

În program se va scrie :

BLOCK CONTAINS 10 RECORDS

sau

BLOCK 10 RECORDS

c. ... *punctele de suspensie*, indică posibilitatea repetării de un număr de ori într-un program a părții precedente din construcția în care apar (în interior sau la sfîrșitul construcției).

Exemplu:

OPEN **INPUT** {nume—fișier}...

ceea ce înseamnă că putem să deschidem în intrare unul sau mai multe fișiere.

Exemplu:

OPEN INPUT FIȘIER—1 FIȘIER—2 FIȘIER—3

Observație

, ; *virgula (sau punctul și virgula)* au caracter opțional și servesc drept separatori (între cuvinte, expresii, clauze, instrucțiuni).

Exemplu:

OPEN INPUT FIȘIER—1; FIȘIER—2

d. Punctul, marchează sfîrșitul unui paragraf, al unei fraze, a unei instrucțiuni condiționale etc.

4.26 Fie instrucțiunea de transfer (atribuire) MOVE cu următoarea structură :

MOVE nume—dată—1 **TO** nume—dată—2 ...

unde :

— MOVE este un cuvînt rezervat cheie și desemnează codul operației ;

— nume—dată—1, va fi înlocuit la scrierea în program cu numele cîmpului (datei) emițător (oare) ;

— nume—dată—2 va fi înlocuit la scrierea în program cu numele cîmpului (datei) receptor (oare)

— punctele de suspensie indică faptul că putem avea mai multe cîmpuri receptoare.

Dacă spre exemplu vrem să scriem operația de transfer a valorii 0 (zero) în cîmpurile CÎMP—1 și CÎMP—2 formulăm instrucțiunea MOVE astfel :

MOVE 0 TO CÎMP—1 CÎMP—2

4.27 *Î.* Punctele de suspensie în formatul descris la paragraful 4.26 pentru instrucțiunea MOVE indică faptul că . . . .

R. numărul cîmpurilor receptoare poate fi mai mare decît 1.

4.29 Forma de prezentare a unei construcții COBOL descrise cu ajutorul simbolurilor speciale și în conformitate cu regulile de alcătuire constituie *formatul general* al construcției respective.

Spre exemplu formatul general al instrucțiunii de transfer (MOVE*) este

$$\textbf{MOVE} \begin{Bmatrix} \text{nume—dată—1} \\ \text{literal} \\ \text{constantă—figurativă} \end{Bmatrix} \quad \textbf{TO} \ \{\text{nume—dată—2}\}\ldots$$

Pentru a asigura o însușire mai rapidă a limbajului COBOL pe tot cuprinsul lucrării pentru aceeași construcție COBOL, se vor prezenta, în general, mai multe variante (sau, cum se menționează în lucrare, *formate*).

4.30 Î. a) Cuvintele rezervate apar în formatele COBOL scrise cu litere . . . . cele cheie sînt . . . . iar cele opționale . . . .

b Cuvintele formate de programator apar în formatele COBOL scrise cu litere . . . . . .

c) Elementele dintre care programatorul, în funcție de cazul concret, alege numai unul, apar în formatele COBOL încadrate de . . . .

d) O construcție opțională (sau utilizabilă numai dacă particularitățile problemei o cer) apare în formatul COBOL încadrată de . . . .

e) Posibilitatea repetării unei construcții COBOL este indicată prin . . . .

f) Cuvintele, expresiile, clauzele, instrucțiunile pot fi separate prin . . . .
iar un paragraf, o frază, o instrucțiune condițională se termină cu . . . .

R. a) mari, mari îngroșate; mari, neîngroșate
b) mici, cursive
c) acolade
d) paranteze pătrate
e) puncte de suspensie (...)
f) virgulă sau punct și virgulă; punct

Răspunsurile sînt corecte?
DA → Lecția a 5-a
NU ↰ paragraful 4.3.

---

* Din motive didactice s-a ales un format general simplificat.

LECȚIA a 5-a STRUCTURA GENERALĂ ȘI COMPONENTELE UNUI PROGRAM COBOL

- structura generală a unui program COBOL
- diviziunea IDENTIFICATION: descriere completă
- diviziunea ENVIRONMENT: structură generală
- formatul simplificat al frazei SELECT
- diviziunea DATA: structură generală
- formatul simplificat al rubricii FD
- formatul simplificat al rubricii de descriere a datelor
- formatul simplificat al diviziunii PROCEDURE
  - instrucțiunea ACCEPT
  - instrucțiunea DISPLAY
  - instrucțiunea STOP RUN

5.1 Un program COBOL este structurat pe patru părți (diviziuni) : IDENTIFICATION DIVISION (diviziunea de identificare) ENVIRONMENT DIVISION (diviziunea de echipament) DATA DIVISION (diviziunea de date; PROCEDURE DIVISION (diviziunea de procedură).

După cum observați denumirile sînt în engleză, deci ele vor fi utilizate ca atare (sînt cuvinte rezervate). Înainte de a trece la un studiu amănunțit al fiecărei diviziuni, indicăm pe scurt conținutul lor. Pentru o mai ușoară înțelegere urmăriți în paralel și programul intitulat ȘCOALA prezentat în lecția 6. În figura 5.1 se prezintă structura generală a unui program COBOL.

5.2 *Diviziunea IDENTIFICATION* furnizează informații în legătură cu numele programului, destinația, funcția și autorul. Ea este constituită numai din paragrafe, dintre care paragraful PROGRAM—ID este obligatoriu (celelalte au rol de comentariu iar semnificația informațiilor conținute este indicată chiar prin titlul paragrafului).

Structura generală a diviziunii IDENTIFICATION este:

```
{IDENTIFICATION}
{ID            }     DIVISION.
PROGRAM—ID.          {nume—program.  }
                     {'nume—program'.}
[AUTHOR.             nume—autor.]
[INSTALLATION.       frază—comentariu.]
[DATE—WRITTEN.       frază—comentariu.]
[DATE—COMPILED.      frază—comentariu.]
[SECURITY.           frază—comentariu.]
[REMARKS.            frază—comentariu.]
```

STRUCTURA GENERALĂ A UNUI PROGRAM COBOL

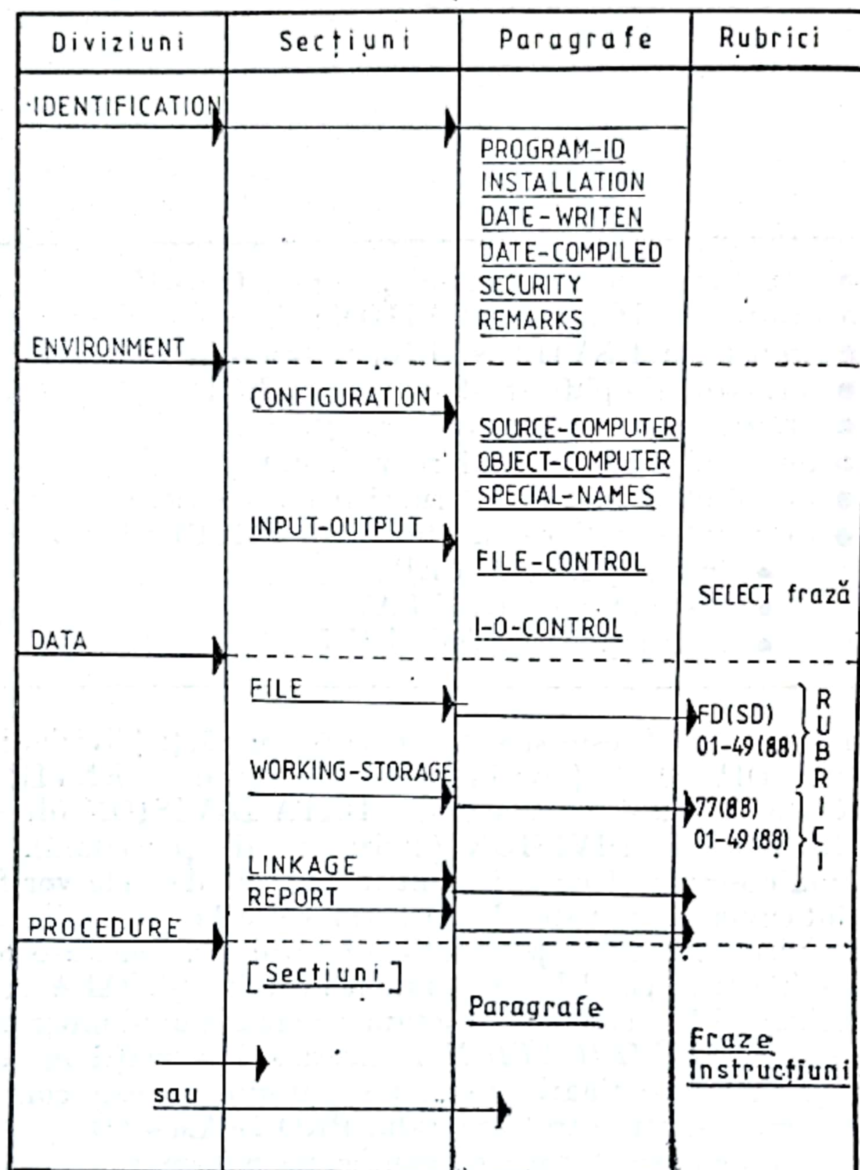


Fig. 5.1.

Exemplu: fig. 5.2.

```
IDENTIFICATION DIVISION
PROGRAM-ID. EXEMPLU.
AUTHOR. POPESCU ION.
```

Fig. 5.2.

Din structura prezentată rezultă că paragraful PROGRAM—ID servește pentru a desemna numele programului, care trebuie, în mod obligatoriu, să înceapă cu o literă și să aibă maximum 8 caractere (fiind un nume extern). Pentru a facilita înțelegerea programului putem utiliza și alte paragrafe sau introduce comentarii folosind linii comentariu (prin marcarea unui asterisc în coloana 7 întreaga linie este considerată comentariu).

Exemple:

a) 7 8
```
 IDENTIFICATION DIVISION.
 PROGRAM—ID. ŞCOALA.
* ACEST PROGRAM REALIZEAZĂ PRELUCRAREA DATELOR
* DIN FIŞIERUL FIŞIER—CARTELE ÎN VEDEREA OBŢINERII
* FIŞIERULUI FIŞIER—BANDĂ
```
b)
```
IDENTIFICATION DIVISION.
PROGRAM—ID. SCOALA.
```

Descrierile *a* şi *b* sînt corecte, iar *a* prezintă în plus şi elemente suplimentare care facilitează înţelegerea programului sau referenţierea particularităţilor sale (7 şi 8 indică coloanele din formularul de programare).

5.3 Î. Diviziunea IDENTIFICATION se compune numai din . . . . . . singurul paragraf obligatoriu fiind . . . .

Numele programului trebuie să înceapă cu . . . . . . şi să aibă maximum . . . caractere. Precizaţi erorile din fig. 5.1. şi 5.2

R. — paragrafe
— PROGRAM—ID
— literă
— 8; WRITTEN cu doi t; după DIVISION se trece caracterul punct.

5.4 Î. Fie o lucrare dată în sarcina programatorului GEORGESCU I. pentru a construi programul numit UPO1.
Programul realizează crearea unui fişier pe bandă magnetică pe baza datelor dintr-un fişier pe cartele.
Fişierul conţinînd date secrete, i se cere programatorului să menţioneze acest lucru în program. Dacă d-voastră aţi fi programatorul GEORGESCU I. cum aţi scrie diviziunea de identificare?
. . . .

R.
```
ID DIVISION.
PROGRAM—ID.    UPO1.
AUTHOR.        GEORGESCU I.
SECURITY.  LA PROGRAM ARE ACCES NUMAI AUTORUL.
REMARKS.   PROGRAMUL CREEAZĂ UN FIŞIER PE BANDĂ
           MAGNETICĂ.
```

Răspunsurile sînt corecte?

NU ↰ paragraful 5.1.
DA → paragraful 5.5.

5.5 *ENVIRONMENT DIVISION*, furnizează informaţii în legătură cu configuraţia, caracteristicile fişierelor utilizate în prelucrare şi modul de tratare a acestora. Ea are două secţiuni: CONFIGURATION SECTION, care conţine paragrafe pentru specificarea configuraţiei şi INPUT—OUT—

— PUT SECTION care conţine paragrafe privind caracteristicile fişierelor utilizate în intrare—ieşire şi modul de tratare a acestora.

5.6 *Secţiunea CONFIGURATION* cuprinde două paragrafe considerate de compilator comentarii (SOURCE—COMPUTER şi OBJECT—COMPUTER) şi paragraful SPECIAL—NAMES în care sînt indicate opţiunile de editare (înlocuirea punctului zecimal cu virgula zecimală şi a simbolului monetar). Toate paragrafele sînt opţionale.

Exemplu:

```
CONFIGURATION SECTION.
SOURCE—COMPUTER. FELIX C—256.
OBJECT—COMPUTER. FELIX C—256.
SPECIAL—NAMES. DECIMAL—POINT IS COMMA.
```

Rezultă că paragraful SOURCE—COMPUTER, indică sistemul de calcul care realizează compilarea programului în timp ce paragraful OBJECT—COMPUTER indică sistemul de calcul care realizează execuţia programului.

Opţiunea DECIMAL—POINT IS COMMA precizează că la editare, punctul zecimal (opţiune implicită) se va înlocui cu virgula zecimală.

5.7 În *secţiunea* INPUT—OUTPUT sînt descrise caracteristicile fişierelor (paragraful FILE—CONTROL) şi modul de tratare a acestora (paragraful I—O CONTROL).

În paragraful FILE—CONTROL pentru fiecare fişier se va utiliza cîte o frază SELECT prin care se indică caracteristicile acestuia.

Structura simplificată a frazei SELECT este:

SELECT nume—fişier **ASSIGN** TO $\left\{\begin{array}{l}\textbf{SYSIN}\\ \textbf{SYSOUT}\\ \textbf{SYSPUNCH}\end{array}\right\}$.

unde:

- *nume-fişier*, reprezintă un nume utilizator atribuit fişierului;
- *ASSIGN*, clauză prin care se precizează identificatorul de exploatare al fişierului *)
- *SYSIN*, cuvînt rezervat care identifică unitatea de intrare standard a sistemului (cititorul de cartele) **);
- *SYSOUT*, cuvînt rezervat care identifică unitatea de ieşire standard a sistemului (imprimanta) **);
- *SYSPUNCH*, cuvînt rezervat care identifică unitatea standard pentru perforare (perforatorul de cartele).

Exemplu:

Pentru un program care utilizează în intrare fişierul pe cartele intitulat FIŞIER—CARTELE iar în ieşire fişierul la imprimantă FIŞIER—LISTA (fig. 5.3.) frazele SELECT au structura:

```
SELECT FIŞIER—CARTELE  ASSIGN TO SYSIN.
SELECT FIŞIER—LISTA    ASSIGN TO SYSOUT.
```

* Orice fişier are un nume care permite referenţierea lui în program şi un identificator de exploatare (idex) folosit de SGF.

** Pentru moment, în scopul simplificării expunerii, se consideră că unitatea standard de intrare este cititorul de cartele iar unitatea standard de ieşire imprimanta.

În lipsa altor clauze în fraza SELECT, compilatorul consideră că fişierul are organizarea secvenţială iar accesul la date este de asemenea secvenţial.

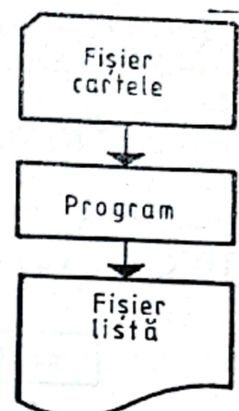


Fig. 5.3.

5.8 *Î.* Rezultă că diviziunea . . . . conţine două . . . . . şi anume . . . . şi . . . . Paragrafele secţiunii CONFIGURATION sînt opţionale, dintre acestea amintim paragraful . . . . şi paragraful . . . .

În cadrul secţiunii INPUT–OUTPUT sînt două paragrafe, şi anume . . . . în care se specifică . . . . şi . . . . în care se specifică . . . .

R. — ENVIRONMENT
— secţiuni
— CONFIGURATION
— INPUT—OUTPUT
— SOURCE—COMPUTER
— OBJECT—COMPUTER
— FILE—CONTROL
— caracteristicile fişierelor
— I—O CONTROL
— modul de tratare a fişierelor.

5.9 *Î.* Pentru fiecare fişier utilizat în program trebuie, în mod obligatoriu, să se descrie caracteristicile acestuia (identificatorul de exploatare, metoda de organizare, modul de acces etc.), cu ajutorul frazei . . . . Dacă în program se utilizează un fişier pe cartele numit MIŞCĂRI şi un fişier la imprimantă numit FIS—SIT frazele SELECT se scriu . . . .

R. SELECT

```
SELECT    MIŞCĂRI    ASSIGN    TO    SYSIN.
SELECT    FIS—SIT    ASSIGN    TO    SYSOUT.
```

5.10 În vederea executării unui program care foloseşte structuri de fişier se cere să se cunoască fişierele utilizate şi zonele de memorie afectate pentru înregistrarea articolelor în memoria centrală.

Aşa cum am văzut în lecţia precedentă descrierea datelor are ca obiectiv să precizeze, în general, natura şi lungimea acestora (incluse sau nu în cadrul unui articol).

Descrierea caracteristicilor generale ale fişierelor utilizate în program, a articolelor asociate acestora şi a celorlalte date folosite în program se realizează în diviziunea DATA.

5.11 *Diviziunea DATA* este structurată pe patru secţiuni:

— *FILE SECTION*, în care sînt descrise fişierele şi articolele corespunzătoare;

— *WORKING—STORAGE SECTION*, în care sînt descrise datele de lucru;

— *LINKAGE SECTION*, care are structura similară cu WORKING—STORAGE, cu menţiunea că aici se descriu datele de legătură în condiţiile în care programul principal foloseşte subprograme *).

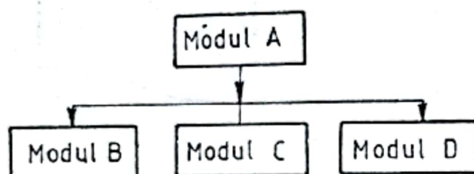


Fig. 5.4.

O problemă de mare complexitate, pentru a fi rezolvată pe calculator se împarte în unităţi mai mici numite module. Modulului Modul—A din figura 5.4 i se va asocia un program principal iar modulelor Modul—B, Modul—C, Modul—D subprograme.

Spre deosebire de modulele B, C şi D care realizează diferite părţi ale lucrării, modulul A este de comandă; el activează modulele B, C şi D şi comunică cu acestea cu ajutorul unor date numite şi *date de legătură*.

Pentru fiecare modul se va concepe cîte un program COBOL. Programul pentru modulul A este de bază (program principal). În cadrul subprogramelor pentru modulele B, C şi D datele de legătură cu modulul A se descriu în secţiunea LINKAGE;

— *REPORT SECTION*, în care sînt descrise situaţiile de obţinut la imprimantă în varianta folosirii editorului COBOL (vezi lecţia 17).

5.12 *Î.* Diviziunea DATA cuprinde un număr de . . . secţiuni şi anume:
. . . .

R. — 4
— FILE—SECTION
— WORKING—STORAGE SECTION
— LINKAGE SECTION
— REPORT SECTION

Remarcă

Secţiunile sînt opţionale, ele se folosesc numai dacă specificul problemei impune acest lucru. Dacă, spre exemplu, scriem un program simplu, în care nu avem date intermediare de lucru (altele decît cele cuprinse în fişiere) este suficientă secţiunea FILE. Se poate ca într-un program să se descrie şi numai secţiunea WORKING—STORAGE.

---

* Într-o formă intuitivă spunem că programul principal realizează funcţii de coordonare-control iar subprogramele funcţii de prelucrare.

5.15 Unităţile structurale ale diviziunii DATA sînt *rubricile de descriere;* ele grupează clauze care caracterizează fişierele sau datele folosite în program. Distingem două tipuri de rubrici:

— rubrici de descriere a fişierelor, care pot apare numai în secţiunea FILE;

— rubrici de descriere a datelor, care pot apare atît în secţiunea FILE pentru a descrie datele organizate în fişiere cît şi în secţiunile WORKING—STORAGE, LINKAGE, REPORT pentru a descrie celelalte date.

Rubrica pentru descrierea fişierelor pe cartele şi la imprimantă are formatul simplificat:

```
FD nume—fişier

  [RECORDING   MODE   IS   {F}]
                           {V}

  LABEL {RECORD    IS }  OMITTED.
        {RECORDS   ARE}
```

Pentru fiecare fişier caracterizat cu ajutorul frazei SELECT se scrie cîte o rubrică FD.

Prin *clauza* RECORDING se precizează formatul articolelor (acesta poate fi fix (F) sau variabil (V)); cînd clauza este omisă se consideră că articolele au lungime variabilă.

Un fişier pe cartele are întotdeauna articole de format fix. Un fişier la imprimantă poate avea articole de format fix sau variabil (de lungime 133 caractere, dintre care primul este de salt).

Clauza LABEL, precizează tipul etichetelor. Din formatul de mai sus rezultă că pentru aceste fişiere etichetele sînt omise (OMITTED).

Exemplu: fie FIŞIER—CARTELE un fişier memorat pe cartele; rubrica FD pentru acest fişier are structura:

```
FD  FIŞIER—CARTELE RECORDING  F
                   LABEL  RECORD  OMITTED.
```

Rubrica pentru descrierea datelor (*structurii articolelor*) are formatul simplificat:

```
număr—de—nivel {nume—dată* }  [{PICTURE}  IS  şablon].
               {FILLER     }   {PIC    }
```

unde:

- *PICTURE* (*PIC*), clauză care precizează prin intermediul şablonului natura şi lungimea unei date elementare;
- *FILLER*, cuvînt rezervat care precizează date care nu se apelează prin program;
- *nume-dată*, cuvînt utilizator format în conformitate cu regulile menţionate în lecţia 4;

* *Nume—dată* este echivalent cu *nume—cîmp* şi *identificator*.

- *şablon*, şir de caractere care precizează natura şi lungimea cîmpului;
- *număr—de—nivel*, grup de maxim două cifre care specifică poziţia ierarhică a datei *nume—dată*.

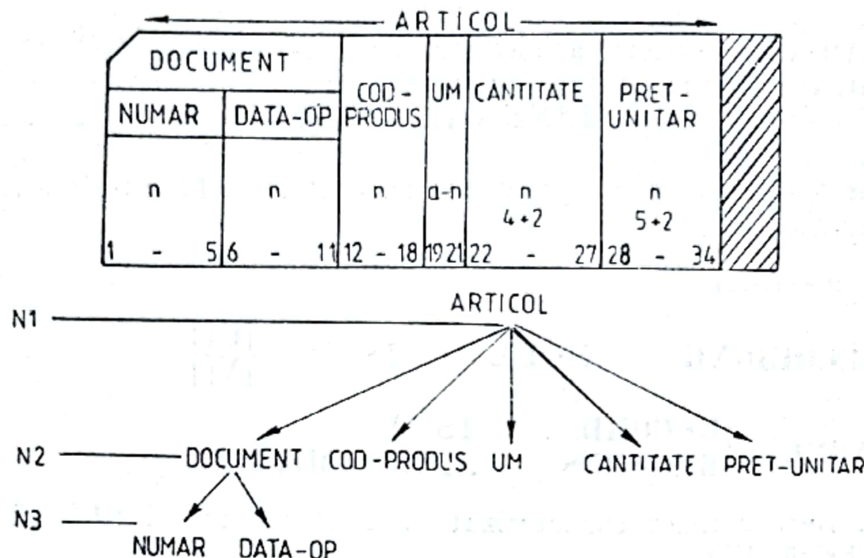


Fig. 5.5.

Spre exemplu, pentru structura articolului prezentată în fig. 5.5. se realizează următoarea descriere:

```
01      ARTICOL.
   02   DOCUMENT.
        03  NUMĂR          PIC 9(5).
        03  DATA—OP        PIC 9(6).
   02   COD—PRODUS         PIC 9(7).
   02   UM                 PIC XXX.
   02   CANTITATE          PIC 9(4)V99.
   02   PRET—UNITAR        PIC 9(5)V99.
```

După cum se poate constata clauza PIC nu apare decît la nivelul rubricilor de descriere a datelor elementare.

5.14 *Î.* Orice rubrică de descriere a datelor începe cu . . . .
şi se termină cu . . . .
Pentru datele elementare structura generală a rubricii de descriere este . . . .

R. — numărul de nivel

— punct

— nr.—de—nivel nume—dată $\begin{Bmatrix}\textbf{PICTURE}\\ \textbf{PIC}\end{Bmatrix}$ şablon.

5.15 *Î.* Pentru a descrie un fişier în cadrul diviziunii . . . .
secţiunea . . . . se utilizează o rubrică . . . .
şi una sau mai multe rubrici . . . .
Rubricile conţin una sau mai multe . . . .

R. — DATA
— FILE
— pentru descrierea fişierului (FD)
— pentru descrierea structurii articolului
— clauze

5.16 Descrierea unui fişier la imprimantă prezintă unele particularităţi, care vor fi prezentate în exemplul care urmează: structura articolului la imprimantă (cu două exemple concrete): este redată în fig. 5.6. unde:

	5	COD-MATERIAL-L	4	CANTITATE-L	4	PRET-L
RIND		12531234 12542436		100.00 200.00		10.00 25.00

Fig. 5.6.

— 5, 4, 4, indică spaţiile lăsate între coloane,
— RÎND, este numele articolului la imprimantă.

O variantă de descriere în program pentru fişierul LISTA obţinut la imprimantă este:

```
FD    LISTA    RECORDING          F
      LABEL    RECORD             OMITTED.
01    RIND.
      02   FILLER                 PIC     X(5).
      02   COD—MATERIAL—L         PIC     9(7).
      02   FILLER                 PIC     X(4).
      02   CANTITATE—L            PIC     Z(4)9.99.
      02   FILLER                 PIC     X(4).
      02   PRET—L                 PIC     ZZ9.99.
```

5.17 Cuvîntul rezervat FILLER este folosit pentru descrierea unui cîmp neutilizat în prelucrare, prin urmare cîmpul nu poate fi referit.

5.18 În şabloanele pentru cîmpurile CANTITATE—L şi PRET—L (formate din întregi şi zecimale) s-a utilizat caracterul punct în loc de V (utilizat numai la descrierea datelor numerice de calcul). Folosirea punctului determină apariţia lui la imprimantă în poziţia precizată în şablon; spunem că se realizează o înserare în procesul de editare.

5.19 Caracterul Z utilizat în descrierea şabloanelor pentru cîmpurile numerice de editare creează posibilitatea eliminării zerourilor nesemnificative.

Aşa cum s-a putut observa din exemplul luat Z-ul a fost utilizat numai pentru cîmpurile CANTITATE—L şi PRET—L, deoarece ele pot avea valori cu un număr variabil de cifre şi există prin urmare posibilitatea apariţiei în partea stîngă (datele numerice se aliniază la dreapta) a zerourilor nesemnificative. Pentru exemplificare, în tabelul 5.1, se prezintă relaţia între conţinut, şablon şi rezultat imprimat pentru un cîmp denumit CANTITATE. Dacă şablonul se scrie Z(5).99 pentru conţinutul cîmpului 00000.75 rezultatul imprimat ar fi fost .75 ceea ce nu corespunde scrierii uzuale.

Tabelul 5.1.

Conținutul posibil al cîmpului (în număr de întregi și zecimale)	Șablon	Rezultat imprimat
2 5 1 5 6 . 1 5	Z(4)9.99	25156.15
0 3 5 4 6 . 2 5	Z(4)9.99	3546.25
0 0 2 3 2 . 1 8	Z(4)9.99	232.18
0 0 0 5 0 . 5 0	Z(4)9.99	50.50
0 0 0 0 8 . 3 0	Z(4)9.99	8.30
0 0 0 0 0 . 7 5	Z(4)9.99	0.75

5.20 *Î.* Vă rugăm să scrieți șabloanele pentru următoarele cîmpuri numerice de editare (cîmpuri care pot avea un număr variabil de cifre) :
CANTITATE (6 + 2) . . . .
STOC (8 + 3) . . . .
PRET (4 + 2) . . . .

R. Z(5)9.99.
Z(7)9.99.
ZZZ9.99.

5.21 *Î.* Formatul simplificat al rubricii pentru descrierea fișierului este : . . . .

R. verificați cu formatul prezentat în paragraful 5.13.

5.22 *Î.* Formatul simplificat al rubricii pentru descrierea structurii articolului este : . . . .

R. verificați cu formatul prezentat în paragraful 5.13.

5.23 *Î.* Clauza PIC servește pentru a preciza cu ajutorul șablonului . . . .

R. natura și lungimea datelor elementare.

5.24 *Î.* Cum apare la imprimantă numărul 1345.66 dacă el reprezintă valoarea unui cîmp numeric de calcul și a fost declarat cu șablonul 9(6)V99 . . . . ; dar dacă ar fi reprezentat valoarea unui cîmp numeric de editare descris cu șablonul Z(5)9.99 . . . . .

R. 00134566
1345.66

5.25 *Î.* Fie un fişier pe cartele numit F—CARTELE căruia i se asociază structura de articol din fig. 5.7.

Descrieţi : a) fraza SELECT
b) rubrica FD
c) rubrica pentru descrierea structurii articolului
d) rubricile pentru descrierea datelor componente

ART-CART

NUMAR MATRICOL (MARCA) 4c	NUMELE SI PRENUMELE 20c	RETRIBUTIA TARIFARA 4c

Fig. 5.7.

. . . .
. . . .
. . . .
. . . .

R. a) SELECT F—CARTELE ASSIGN SYSIN.
b) FD F—CARTELE RECORDING F LABEL RECORD OMITTED.
c) 01 ART—CART.

```
        02  MARCA     PIC   9(4).
        02  N—P       PIC   A(20).
        02  R—T       PIC   9(4).
```

5.26 *Diviziunea PROCEDURE* grupează instrucţiunile care traduc în comenzi COBOL algoritmul lucrării şi are formatul simplificat :

PROCEDURE DIVISION.
{[nume—secţiune **SECTION.**]
{nume—paragraf.
{frază.}...}...}...

Rezultă că diviziunea PROCEDURE poate conţine secţiuni ; numele şi numărul acestora fiind stabilit de programator (se indică secţionarea diviziunii PROCEDURE numai în cazul programelor complexe şi cînd cerinţele o impun).

În cazul oricărui program, diviziunea PROCEDURE trebuie să înceapă în mod obligatoriu cu un nume de paragraf (sau de secţiune).

Frazele conţin una sau mai multe instrucţiuni COBOL şi se termină cu punct urmat de spaţiu.

Instrucţiunile COBOL permit prelucrarea datelor conform algoritmului lucrării.

Aşa cum o să vedem în cele ce urmează există două categorii de instrucţiuni şi anume :

— *instrucţiuni imperative*, care se execută necondiţionat (transfer, adunare, scădere etc.) ;

— *instrucțiuni condiționale* care permit realizarea operațiilor de testare a unor condiții sau de executare condiționată a unor operații.

5.27 *Î.* Diviziunea PROCEDURE poate avea a) . . . . și în mod obligatoriu trebuie să înceapă cu un nume de b). . . . .

R. a) secțiuni
paragrafe
b) paragraf sau de secțiune

5.28 Cele mai simple instrucțiuni ale diviziunii de procedură sînt:

— ACCEPT, pentru introducerea unor date neorganizate în fișiere (în general folosind cititorul de cartele sau consola);

— DISPLAY, pentru afișarea unor rezultate la imprimantă sau la consolă fără o organizare a acestora în fișiere;

— STOP RUN, pentru oprirea execuției programului.

5.29 Programul 5.1 citește un text dintr-o cartelă și-l imprimă ca atare.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROGRAM1.
*
**   ACEST PROGRAM ESTE FOARTE SIMPLU
**   SI ARE ROL DIDACTIC
**
AUTHOR. POPESCU ION.
ENVIRONMENT DIVISION.
*
CONFIGURATION SECTION.
SOURCE-COMPUTER. FELIX C-256.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  CARTELA.
  02 NUMAR            PIC 9(6).
  02 FILLER           PIC X(74).
PROCEDURE DIVISION.
*
INCEPUT.
    ACCEPT CARTELA
    DISPLAY CARTELA
    STOP RUN.
                          PROGRAMUL 5.1
```

Programul 5.1.

5.30 *Instrucțiunea ACCEPT* are formatul simplificat:

ACCEPT nume—dată [**FROM CONSOLE**]

unde:
- *nume—dată* reprezintă un cîmp elementar sau grup descris în secțiunea WORKING—STORAGE;
- *FROM CONSOLE*, precizează că datele sînt introduse de către operator de la mașina de scris (această opțiune este mai puțin indicată).

Instrucţiunea DISPLAY are formatul simplificat:

DISPLAY {literal | nume—dată} [**UPON CONSOLE**]

unde:
- *nume—dată* reprezintă un cîmp elementar sau grup;
- *UPON CONSOLE*, precizează că operaţiile de înregistrare se realizează la maşina de scris.

5.31 𝒥. Scrieţi un program COBOL care-şi propune să citească două cartele ce conţin date omogen structurate (ca în fig. 5.8) şi să afişeze conţinutul acestora.

ℛ. Programul 5.2. din pag. 90

5.32 Întrucît programul prezentat a fost rulat pe calculator sînt necesare unele explicaţii referitoare la etapele tratării acestuia. Pentru mai multe detalii cititorul va apela la lecţia 16. Intenţionat programul 5.2 a fost scris cu erori (în liniile 100 şi 220 nu s-a marcat punctul după A şi respectiv N). La perforare deasemenea pot apare erori.

COD-MATERIAL n 7	CANTITATE n 5	PRET-UNITAR n 3	

Fig. 5.8.

Pentru a fi executabil un program parcurge următoarele faze:

— *compilarea*, care are ca rezultat obţinerea modulelor în BT (binar translatabil). În urma acestei faze programatorul primeşte un listing (hîrtie de imprimantă) care conţine liniile programului sursă şi, dacă au fost erori, mesajele privind natura acestora (program 5.3, din p. 91);

— *linkeditarea*, realizarea legăturilor între modulele BT în vederea obţinerii programului unitar în format IMT (imagine memorie translatabilă). Pentru realizarea acestei faze se foloseşte cartela de comandă LINK. Programul obţinut *) este memorat pe discul sistem (discul care conţine programele sistemului de operare);

— *executarea*, fază în care programul este încărcat de pe discul sistem în memoria centrală şi lansat în execuţie. În acest scop se adaugă cartelele de comandă RUN şi cartelele de date.

Programul intitulat PROGRAM2 în varianta fără erori furnizează, în urma editării legăturilor şi executării, informaţiile din programul 5.4. (pag. 92-94) (pentru eliminarea erorilor în linia sursă 10 s-a trecut caracterul „—" după litera D, iar în linia 20 s-a trecut punctul terminal după litera N).

Răspunsurile au fost corecte?

DA → Lecţia 6
NU ↰ 5.1.

---

* Pentru început, cititorul neavizat trebuie să înţeleagă că în urma compilării nu rezultă un program executabil (chiar dacă nu sînt erori). Cu ajutorul comenzii LINK adresate sistemului de operare, se aduce programul în format executabil.

UNITATEA
CONTUL

PROGRAM COBOL

LUCRAREA	PROGRAMUL	INSTRUCȚIUNI DE PERFORARE	SIMBOLUL	PAG. DIN
PROGRAMATOR	DATA		SE PERFOREAZĂ	73 IDENTIF. 80

SECVENȚA — INSTRUCȚIUNILE PROGRAMULUI

```
010 ID DIVISION.
020*
030 PROGRAM-ID. PROGRAM2.
040 ENVIRONMENT DIVISION.
050*
060 CONFIGURATION SECTION.
070 DATA DIVISION.
080*
090 WORKING-STORAGE SECTION.
100 01 CARTELA.
110     02 COD-MATERIAL          PIC 9(7).
120     02 CANTITATE             PIC 9(5).
130     02 PRET-UNITAR           PIC 999.
140*
150 PROCEDURE DIVISION.
160*
170 INCEPUT.
180        ACCEPT CARTELA
190        DISPLAY CARTELA
200        ACCEPT CARTELA
210        DISPLAY CARTELA
220        STOP RUN.
230*
240
250
```

Programul 5.2.

```
   JOB FLOR1610,AN:FPFD,PN:FLORESCU,CLS:1
   COMPILE COBOL
• OBOL      STARTED

         0001               ID DIVISION.                                       COBOL*
       C 0002              *
         0003               PROGRAM-ID. PROGRAM2.
         0004               ENVIRONMENT DIVISION.
       C 0005              *
         0006               CONFIGURATION SECTION.
         0007               DATA DIVISION.
         0008               WORKING-STORAGE SECTION.
         0009               01  CARTELA.
         0010                 02 COD MATERIAL   PIC 9(7).
         0011                 02 CANTITATE      PIC 9(5).
         0012                 02 PRET-UNITAR    PIC 999.
         0013               PROCEDURE DIVISION.
       C 0014              *
         0015               INCEPUT.
         0016                   ACCEPT CARTELA
         0017                   DISPLAY CARTELA
         0018                   ACCEPT CARTELA
         0019                   DISPLAY CARTELA
         0020                   STOP RUN
                                                                                  COBOL*
                                            ETAT RECAPITULATIF DU PROGRAMME: PROGRA
 SEGMENT PROGR%00         NUMERO 01

      MODULE D PROGRA00       LONGUEUR 0130

      MODULE P PROGRA01       LONGUEUR 0048


      FIN NORMALE DE COMPILATION


 ETAT  DES  DIAGNOSTICS


• RAVITE MAXIMALE ADMISE :    2.
• OMBRE D'ERREURS DE GRAVITE  2:     1
• OMBRE D'ERREURS DE GRAVITE  3:     1

                                                                                  COBOL
• LSR  COL MESS.  GRAV  LIBELLE DU MESSAGE


   10  24   3003  G:2*  DESCRIPTION DE DONNEE ERRONEE OU POINT TERMINAL ABSENT
   20  19   4063  G:3*  ABSENCE DE POINT OU ERREUR DE SYNTAXE


• IN DE COMPILATION COBOL PASSAGE A LA PHASE SUIVANTE


• EURE FIN : 09H52M34
```

Programul 5.3.

• Atenţie! La imprimantă s-a tipărit slab primul caracter din rîndurile însemnate cu • vom citi succesiv COBOL, GRAVITE, NOMBRE, NOMBRE, NLSR, FIN, HEURE.

```
        JOB FLOR1610,AN:FPFU,PN:FLORESCU,CLS:1
        COMPILE COBOL
COBOL   STARTED

               ID DIVISION.
              *
               PROGRAM-ID. PROGRAM2.
               ENVIRONMENT DIVISION.
              *
               CONFIGURATION SECTION.
               DATA DIVISION.
               WORKING-STORAGE SECTION.
               01  CARTELA.
                   02  COD-MATERIAL       PIC 9(7).
                   02  CANTITATE          PIC 9(5).
                   02  PRET-UNITAR        PIC 999.
               PROCEDURE DIVISION.
              *
               INCEPUT.
                     ACCEPT CARTELA
                     DISPLAY CARTELA
                     ACCEPT CARTELA
                     DISPLAY CARTELA
                     STOP RUN.
              *                        PROGRAMUL 5,2

                                 ETAT RECAPITULATIF DU PROGRAMME:
SEGMENT PROGR%00         NUMERO 01

     MODULE 0 PROGRA00        LONGUEUR 0130

     MODULE 2 PROGRA01        LONGUEUR 0048


FIN NORMALE DE COMPILATION


 E T A T   D E S   D I A G N O S T I C S


GRAVITE MAXIMALE ADMISE :    2.


AUCUNE ERREUR
```

Programul 5.4.

```
FIN DE COMPILATION COBOL PASSAGE A LA PHASE SUIVANTE -

HEURE FIN : 09H51M11
```

```
CENTRUL DE CALCUL ASE BUCURESTI                                   SYSTEM - 015
0015  FLORI610    AN = FFFU    PH = 0001    DATE = 16/06/81.167
H.DEB = 09H 30M 52S    H.FIN = 09H 31M 13S    TIME = 00000297    CODE = 000
LGP = 00060    MEM = 00015    LO = 000056    IN = 00025    OUT = 00000
```

Programul 5.4. (continuarea I)

```
           LINK
LINK      STARTED


LINK      15.06.01    16/06/81    09H31M27S


SEGMENT   PROGRX00  NO    1    IMPLANTATION      0
          MODULE  PROGRA01     IMPLANTATION     38
          MODULE  TDFSYSIN     IMPLANTATION     80
          MODULE  TDFSYSUT     IMPLANTATION     F8
          MODULE  C87          IMPLANTATION    168
          MODULE  C76          IMPLANTATION    230
          MODULE  STOPRUN      IMPLANTATION    320
          MODULE  ACCEPT       IMPLANTATION    390
          MODULE  C75A         IMPLANTATION    300
          MODULE  C%WRITE2     IMPLANTATION    4A8
          MODULE  OPCLTYP      IMPLANTATION    500
          MODULE  C%WRITEA     IMPLANTATION    610
          MODULE  C%WRITEB     IMPLANTATION    648
                                      LONGUEUR DU SEGMENT  6A0

SEGMENT   PROGRX00  NO    2    IMPLANTATION    6A0
          MODULE  PROGRA00     IMPLANTATION      0
                                      LONGUEUR DU SEGMENT  130
```

Programul 5.4. (continuarea II)

```
LINK      15.06.01    16/06/81   09H31M27S

                       IMPLANT. APRES IRAITEMENT  PTION FMS

   SEGMENT    PROGRX00  NO   1    IMPLANTATION     0
                                            LONGUEUR DU SEGMENT 2228

   SEGMENT    PROGRX00  NO   2    IMPLANTATION  2228
                                            LONGUEUR DU SEGMENT  130


LINK      15.06.01    16/06/81   09H31M27S

            0      ERREUR  EN EDITION DE LIENS
          ADRESSE DE LANCEMENT                     36
          LONGUEUR PLUS GRANDE BRANCHE           2358
          LONGUEUR DU PROGRAMME EDITE            2356

                PLUS HAUT NIVEAU D'ERREUR RENCONTRE  N=0    (PAS D'ERREUR)


            CENTRUL DE CALCUL ASE BUCURESII                         SYSTEM - 813
            0015  FLORI610     AN = FPFO    PH = 0002   DATE = 16/06/81-167
            H.DEB = 09H 31M 13S    H.FIN = 09H 31M 29S   TIME = 00000350   CODE = 000
            LGP = 00060    MEM = 00012   LO = 000040   IN = 00000    OUT = 00000


   .       RUN.
          STARTED

  135676100345010
  135678200035001


              CENTRUL DE CALCUL ASE BUCURESTI                        SYSTEM - 813
              0015  FLORI610   AN = FPFO   PH = 0003   DATE = 16/06/81-167
              H.DEB = 09H 31M 30S   H.FIN = 09H 31M 34S   TIME = 00000080   CODE = 000
              LGP = 00060   MEM = 00005   LO = 000008   IN = 00003   OUT = 00000
```

Programul 5.4. (continuarea III)

LECȚIA a 6-a INSTRUCȚIUNI DE LUCRU CU FIȘIERE MEMORATE PE CARTELE SAU HÎRTIE DE IMPRIMANTĂ

- instrucțiunea OPEN
- instrucțiunea CLOSE
- instrucțiunea MOVE
- instrucțiunea READ
- instrucțiunea WRITE
- instrucțiunea GO TO
- instrucțiunea PERFORM ... UNTIL
- scrierea în formularul de programare
- program cu fișiere: caz simplificat
- reguli de punctuație
- cartele de comandă

6.1 Principalele instrucțiuni utilizate în diviziunea de procedură pentru prelucrarea datelor organizate în fișiere sînt:

— *OPEN*, pentru descrierea operațiilor de deschidere a fișierelor;

— *CLOSE*, pentru descrierea operațiilor de închidere a fișierelor;

— *MOVE*, pentru descrierea operațiilor de transfer a datelor în interiorul memoriei centrale;

— *READ*, pentru descrierea operațiilor de citire a unui articol dintr-un fișier;

— *WRITE*, pentru descrierea operațiilor de scriere (înregistrare) a unui articol într-un fișier;

— *GO TO*, pentru descrierea operațiilor de transfer a controlului într-un anumit punct (paragraf) din cadrul programului;

— *PERFORM ... UNTIL*, pentru descrierea operațiilor de transfer a controlului cu revenire la instrucțiuni imediat următoare (conform structurii repetitive prezentate în lecția 3).

6.2 *Instrucțiunea OPEN*, descrie operația de deschidere a fișierelor prin care se realizează pregătirea în vederea prelucrării și are următorul format simplificat:

$$\textbf{OPEN} \left\{ \begin{Bmatrix} \textbf{INPUT} \\ \textbf{OUTPUT} \end{Bmatrix} \quad \text{nume—fișier} \right\} \ldots$$

unde:

INPUT, opțiune care indică deschiderea în intrare a fișierului;
OUTPUT, opțiune care indică deschiderea în ieșire a fișierului.

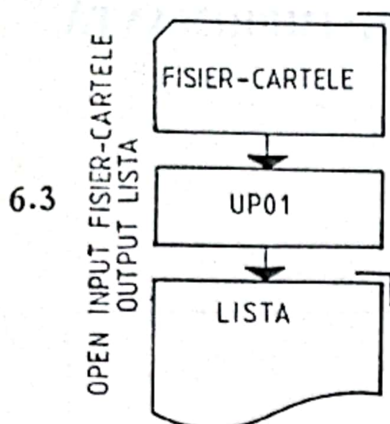


Fig. 6.1.

Exemplu: (fig. 6.1.).

Observație

Pentru fișierele pe suporturi magnetice instrucțiunea OPEN asigură completarea cu anumite date și controlul etichetelor în funcție de tipul prelucrării (creare sau consultare).

6.3 *Instrucțiunea CLOSE*, descrie operația de închidere a fișierului și are formatul simplificat:

CLOSE nume—fișier—1 {nume—fișier—2}...

Exemplu:

CLOSE FIȘIER—INTRARE
FIȘIER—IEȘIRE

Observație

Un fișier utilizat în program trebuie deschis înainte de a fi supus prelucrării și închis după aceea.

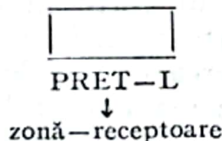
6.4 *Instrucțiunea MOVE* realizează operațiile de transfer și are formatul simplificat:

MOVE $\left\{\begin{matrix}\text{nume—dată—1}\\ \text{literal}\end{matrix}\right\}$ **TO** {nume—dată—2} ...

Exemplu:

MOVE PREȚ TO PREȚ—L





PREȚ ↓ zonă—emițătoare

PREȚ—L ↓ zonă—receptoare

Instrucțiunea MOVE nu distruge conținutul cîmpului emițător. Cînd trebuie să scrieți o listă cu mai multe nume obișnuiți să le aranjați de așa manieră încît prima literă să înceapă pe aceeași linie verticală.

POPESCU
VOICULESCU
GEORGESCU
IONESCU
VOINESCU

În acest caz spunem că ele au fost cadrate la stînga:

starea inițială:

|P|O|P|E|S|C|U| | | | | | | | | | | | |

↑ ZONA—1   ↑ ZONA—2

MOVE ZONA—1 TO ZONA—2

starea finală:

|P|O|P|E|S|C|U| | | |P|O|P|E|S|C|U| | |

Pozițiile din dreapta rămîn la valoarea spațiu:

starea inițială:

```
|I|O|N|E|S|C|U| | |          |7|3|4|M|I|R|C|E|A|1|2|3|
 ↑                            ↑
    ZONA—1                   ZONA—2

         MOVE ZONA—1  TO  ZONA—2
```

starea finală:

```
|I|O|N|E|S|C|U| | |       |I|O|N|E|S|C|U| | | | | |
```

Cînd cîmpul receptor este mai mic decît cîmpul emițător se pierd caracterele cele mai din dreapta (are loc o trunchiere):

starea inițială

```
   ZONA—1               ZONA—2
    |P|O|P|E|S|C|U|         | | | | | |
   MOVE    ZONA—1    TO    ZONA—2
```

starea finală:

```
|P|O|P|E|S|C|U|          |P|O|P|E|S|
```

Același lucru este valabil și pentru datele alfanumerice. Pentru datele de natură numerică alinierea se va face de la dreapta spre stînga și în funcție de punctul zecimal:

```
  34.53|
 156.61|
2345.72|
  15.13↓
```

Cazul cînd ZONA—1 = ZONA—2:

```
starea inițială      |0|0|0|3|5|4|3|        |0|0|0|0|0|0|0|
                              ↑                       ↑
                     ZONA—1            ZONA—2

                     MOVE   ZONA—1  TO  ZONA—2

starea finală:       |0|0|0|3|5|4|3|        |0|0|0|3|5|4|3|
                              ↑                       ↑
```

Cazul cînd ZONA—1 < ZONA—2

starea inițială:

```
                 |1|3|5|4|3|          | | | | | | | | | |
                        ↑
              ZONA—1                     ZONA—2

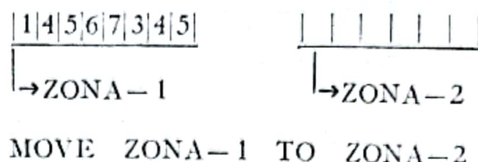
                    MOVE   ZONA—1  TO  ZONA—2
```

starea finală:

```
                 |1|3|5|4|3|          |0|0|0|0|1|3|5|4|3|
```

Cazul cînd ZONA—1 > ZONA—2 :

stare inițială:



starea finală:



6.5 La realizarea operațiilor de transfer folosind instrucțiunea MOVE trebuie avută în vedere natura datelor implicate. Astfel distingem:

a) *transferul numeric*, cînd cîmpul receptor conține date de natură numerică sau numerică de editare.

În acest caz regulile de realizare a operațiilor de transfer sînt:

— în cîmpul receptor datele sînt cadrate în funcție de marca zecimală (pozițiile rămase libere în partea din stînga sînt completate cu zerouri) iar cifrele care depășesc lungimea zonei receptoare se trunchiază (se neglijează);

— în cazul în care formatul datelor din cele două zone este diferit, operația de transfer este precedată de cea de conversie pentru a trece numărul în formatul zonei receptoare;

— în cazul în care cîmpul receptor este descris cu șablon de editare în clauza PICTURE operația de transfer include și editarea.

b) *transferul alfanumeric*, cînd cîmpul receptor conține date de natură alfanumerică, alfabetică sau este un cîmp grup.

În acest caz regulile de realizare a operației de transfer sînt:

— în cîmpul receptor datele sînt cadrate de la stînga la dreapta, cu excepția cazului în care la descrierea cîmpului de dată s-a folosit opțiunea JUSTIFIED*);

— în cazul în care cîmpul receptor are lungimea mai mare decît cel emițător locațiile libere sînt încărcate cu spații, în caz contrar se realizează trunchierea (omiterea caracterelor din dreapta);

— cînd cîmpul emițător conține date numerice în format binar sau zecimal-împachetat se realizează mai întîi conversia în zecimal-despachetat.

Dacă atît cîmpul emițător cît și cel receptor sînt elementare spunem că transferul este elementar.

Dacă cel puțin unul din cîmpuri (emițător sau receptor) este grup, transferul este neelementar și se execută după regulile transferului alfanumeric (fără a se executa conversia datelor dintr-o formă în alta).

Pentru transferul neelementar nu se verifică concordanța dintre natura datelor introduse în cîmpul receptor și natura cîmpului receptor stabilită prin clauza PICTURE.

---

* Clauza JUSTIFIED precizează înregistrarea în cîmpul receptor de la dreapta la stînga.

În tabelul 6.2 se prezintă transferurile permise și nepermise în limbajul COBOL.

Tabelul 6.2.

Cîmp emițător \ Cîmp receptor		a	b	c	d
Cîmp grup	a	A	A	A*	A*
Cîmp elementar — nenumeric	b	A	A	A*	A*
— numeric de calcul	c	A	A_c	N_c	E_c
— numeric de editare	d	A	A	interzis	

unde:

A , transfer numeric;
A*, transfer alfanumeric cu rezultat imprevizibil cînd cîmpul emițător nu conține numere cu semn;
A_c, transfer alfanumeric cu conversie de caractere;
N_c, transfer numeric cu conversie;
E_c, transfer de editare cu inserții și substituiri.

6.6 *Instrucțiunea READ*, are formatul simplificat:

READ nume—fișier AT **END** instrucțiuni ....

Instrucțiunea READ se utilizează pentru consultarea secvențială a unui fișier deschis cu opțiunile INPUT sau INPUT—OUTPUT ale instrucțiunii OPEN.

La executarea instrucțiunii READ primul articol din fișier este transferat în zona articol; articolul rămîne în zona respectivă pînă se realizează o nouă citire.

Cînd s-a detectat marca de sfîrșit fișier se execută instrucțiunea (instrucțiunile) precizate după clauza AT END.

Instrucțiunea READ este deci o instrucțiune condițională și se termină cu punct.

La detectarea mărcii de sfîrșit fișier se asigură și controlul etichetelor standard (pentru fișiere pe suporturi magnetice).

Exemplu:

```
READ FIŞIER—CARTELE AT END
                        MOVE 1 TO VEOF.
```

unde:

— VEOF, este un cîmp care prin valoarea 1 precizează că au fost citite toate articolele din fișierul FIȘIER—CARTELE și s-a detectat marca EOF.

6.7 *Instrucțiunea WRITE* se folosește pentru scrierea unui articol într-un fișier la imprimantă și are următorul format simplificat:

WRITE nume—articol **AFTER** ADVANCING întreg LINES

unde: *opțiunea ADVANCING* indică condițiile de scriere a articolelor, adică după avansarea cu valoarea lui *întreg* de linii, iar întreg poate lua valorile:
0, pentru a preciza saltul la o nouă pagină;
1, pentru a preciza că hîrtia avansează cu un rînd;
2, pentru a preciza că hîrtia avansează cu două rînduri, ş.a.m.d.

Exemple:

```
WRITE RIND AFTER 0
WRITE RIND AFTER 1
```

unde:
RIND, numele articolului asociat fişierului la imprimantă.

6.8 *Instrucțiunea PERFORM ... UNTIL ...* permite descrierea în instrucțiune COBOL a structurii repetitive (ciclului) prezentat în figura 6.2.

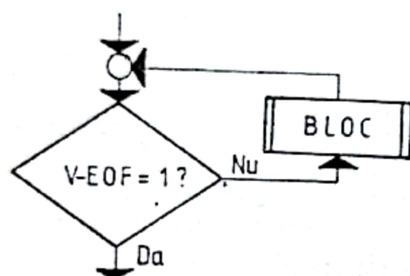


Fig. 6.2.

Din figură rezultă că operațiile desemnate prin nume—prelucrare BLOC sînt executate pînă cînd valoarea condiției V-EOF = 1 este „adevărat".

6.9 Ordinea de realizare a operațiunilor în cazul instrucțiunii PERFORM...UNTIL este următoarea:

— se evaluează condiția;

— atîta timp cît valoarea logică a condiției (care de obicei este un test de relație) este „fals" se execută instrucțiunile din procedură;

— dacă valoarea logică a condiției este „adevărat" se iese din ciclu și se revine la instrucțiunea imediat următoare instrucțiunii PERFORM din program.

Exemplu: Pentru schema logică din figura 6.3. diviziunea de procedură se scrie:

```
PROCEDURE DIVISION.
PROCEDURA—PRINCIPALĂ.
        OPEN INPUT FIŞIER—CARTELE
                OUTPUT FIŞIER—IMPRIMANTĂ
        MOVE 0 TO V—EOF
        READ FIŞIER—CARTELE
                        AT END MOVE 1 TO V—EOF.
        PERFORM PROCEDURA UNTIL V—EOF = 1
        CLOSE  FIŞIER—CARTELE
               FIŞIER—IMPRIMANTĂ
        STOP RUN.
```

```
PROCEDURA.
          MOVE ARTICOL-CARTELA TO ARTICOL-IMPRIMANTA
          WRITE ARTICOL-IMPRIMANTA AFTER 1
          READ FIŞIER-CARTELE AT END
                              MOVE 1 TO V-EOF.
```

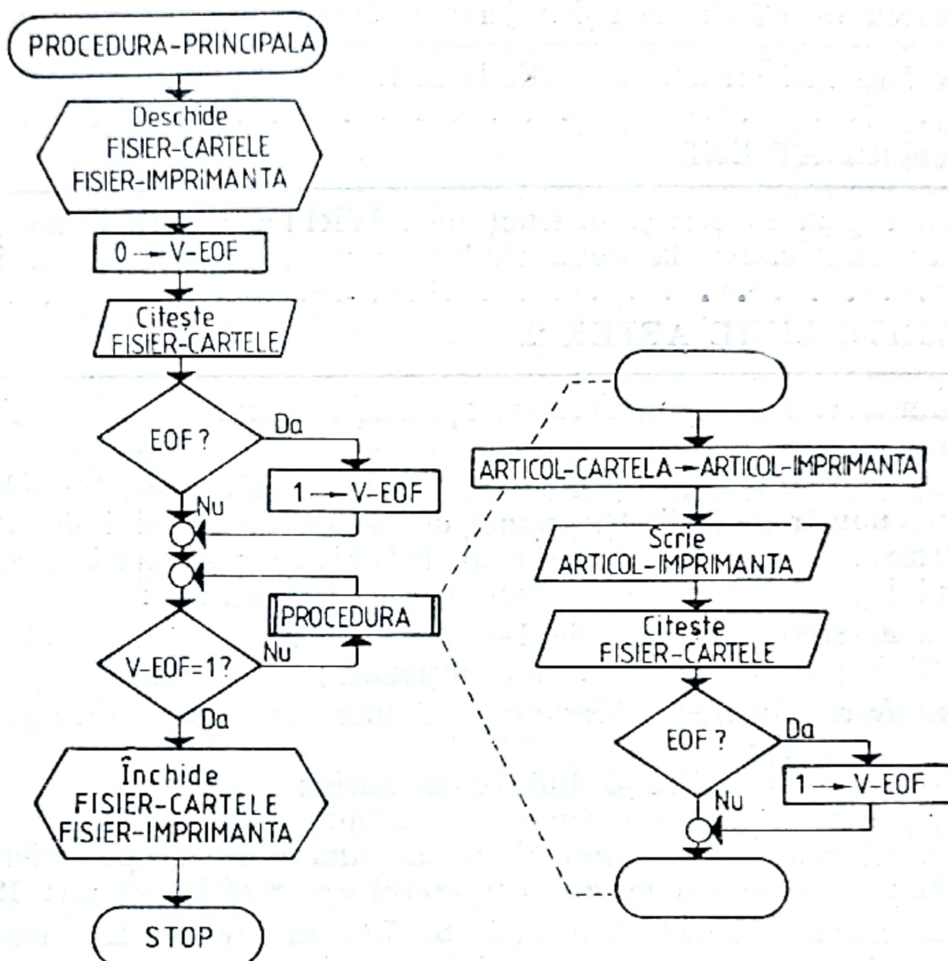


Fig. 6.3.

6.10 *Instrucţiunea GO TO* realizează operaţia de salt şi are formatul simplificat:

GO TO paragraf

6.11 Realizarea saltului într-un anumit punct al programului se face prin instrucţiunea GO TO.

Exemplu:

```
CALCUL.
     instrucţiunea — 1
     instrucţiunea — 2.
REDARE.
     instrucţiunea — 3
```

instrucţiunea — 4
instrucţiunea — 5
GO TO CALCUL.

6.12 Î. Ce reprezintă ultima cartelă dintr-un fişier memorat pe caretele . . . .

R. marca de sfîrşit de fişier (marca EOF)

6.13 Î. Ce face ca instrucţiunea READ să fie o instrucţiune condiţională? . . . .

R. condiţia AT END

6.14 Î. Vă rugăm să scrieţi instrucţiunea WRITE pentru cazul cînd se cere imprimarea la două rînduri, numele rîndului fiind LINIE. . . . .

R. WRITE LINIE AFTER 2.

6.15 Programele se scriu pe formulare speciale structurate în zone, rînduri şi coloane (fig. 6.4).

Formularul COBOL are o liniatură specială: el conţine 80 coloane şi un anumit număr de linii. Din punct de vedere al modului de utilizare, şi prin urmare al semnificaţiei ce o au în scrierea programelor, coloanele formularului de programare se grupează în patru zone:

— *zona de secvenţă* (coloanele 1—6) în care se trece numărul de secvenţă al liniei; completarea ei este opţională;

— *zona de continuare şi delimitare* a liniilor comentariu care poate lua valorile (coloana 7):

* pentru a indica o linie comentariu;
—, pentru a indica o linie ce continuă linia precedentă (pentru linia de continuare a unui literal nenumeric se începe scrierea literalului după ce s-a marcat caracterul apostrof în coloana 12);

— *zona textului sursă* (coloanele 8—72) în care se trec enunţurile COBOL; ea se împarte la rîndul ei în două zone;
— *Zona A*, care începe din coloana 8 şi se foloseşte la scrierea componentelor principale ale programului (diviziunile, secţiunile, paragrafele, rubricile FD şi rubricile de descriere a datelor de nivel 01);
— *Zona* B, care începe din coloana 12 şi se foloseşte pentru scrierea instrucţiunilor din diviziunea de procedură şi a celorlalte componente;

— *zona de identificare* (73—80) unde se trece de obicei numele programului.

6.16 Despărţirea unui cuvînt COBOL altul decît un literal nenumeric este marcată prin '—' în coloana 7 a liniei următoare; continuarea se realizează începînd cu coloana 12.

6.17 Î. În formularul de programare COBOL diviziunile, secţiunile şi paragrafele, rubricile FD şi rubricile de descriere a datelor 01 se scriu începînd din coloana . . . . iar celelalte din coloana . . .

Unitatea ____________

Contul

PROGRAM COBOL

LUCRAREA	PROGRAMUL	INSTRUCTIUNI DE PERFORARE	SIMBOLUL SE PERFOREAZA	Pag. Din
PROGRAMATOR	DATA			73 IDENTIF. 80

SECVENTA (1–6)	CONTINUARE (7)	8–11	INSTRUCTIUNILE PROGRAMULUI (12–72)
010			
020			
030			
040			
050			
060			
070			
080			
090			
100			
110			
120			
130			
140			
150			
160			
170			
180			
190			
200			
210			
220			
230			
240			
250			

Fig. 6.4.

R. 8
12

6.18 Pentru fiecare rînd (linie) din formularul COBOL se perforează cîte o cartelă.

6.19 *Î.* Cînd un literal nenumeric nu poate fi scris în întregime pe un rînd se procedează astfel:

. . . .

R. — se trece liniuță în coloana 7;
— se trece apostroful în coloana 12;
— se continuă literalul.

6.20 Pentru a vedea mai concret structura simplificată a unui program COBOL, vom lua un caz de listare a conținutului unui fișier pe cartele (fig. 6.5).

6.21 Se citesc succesiv articolele din fișierul de pe cartele și se listează la imprimantă.

Operațiile *citește* și *scrie* se repetă de atîtea ori cîte articole sînt în fișierul folosit în intrare. Cînd a fost detectată marca de sfîrșit de fișier (cartela EOF) se consideră fișierul citit în întregime și se poziționează o variabilă booleană numită IND—SF la valoarea DA.

6.22 Structura articolului asociat fișierului F—CARTELE este prezentată în figura 6.6.

6.23 Pe baza datelor descrise în program, memoria afectată se va organiza ca în fig. 6.7.

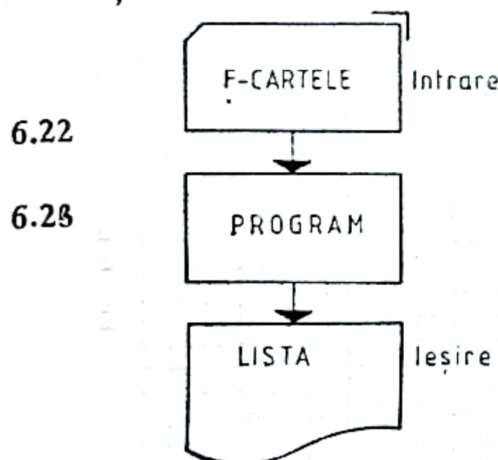


Fig. 6.5.

CODUL MATERIALULUI	CANTITATEA	PREȚUL UNITAR
n 7	n 5+2	n 3+2

Fig. 6.6.

6.24 Schema logică de programare, în varianta programării clasice, este cea din fig. 6.8.

6.25 *Î.* Ce tipuri de structuri s-au utilizat în această schemă logică?

. . . .

R. — structura alternativă (selecție dublă);
— structura repetitivă în forma cu testarea condiției după ce s-a început execuția procedurii (fig. 6.9).

Varianta schemei logice care respectă conceptele programării structurate este redată în fig. 6.10. (variabilei SFÎRȘIT-FIȘIER din schemă îi corespunde în program variabila IND-SF).

6.26 Așa cum rezultă din fig. 6.7 datele pentru a fi supuse prelucrării trebuie aduse în memoria centrală (prin comanda citește). Articolul citit se depune

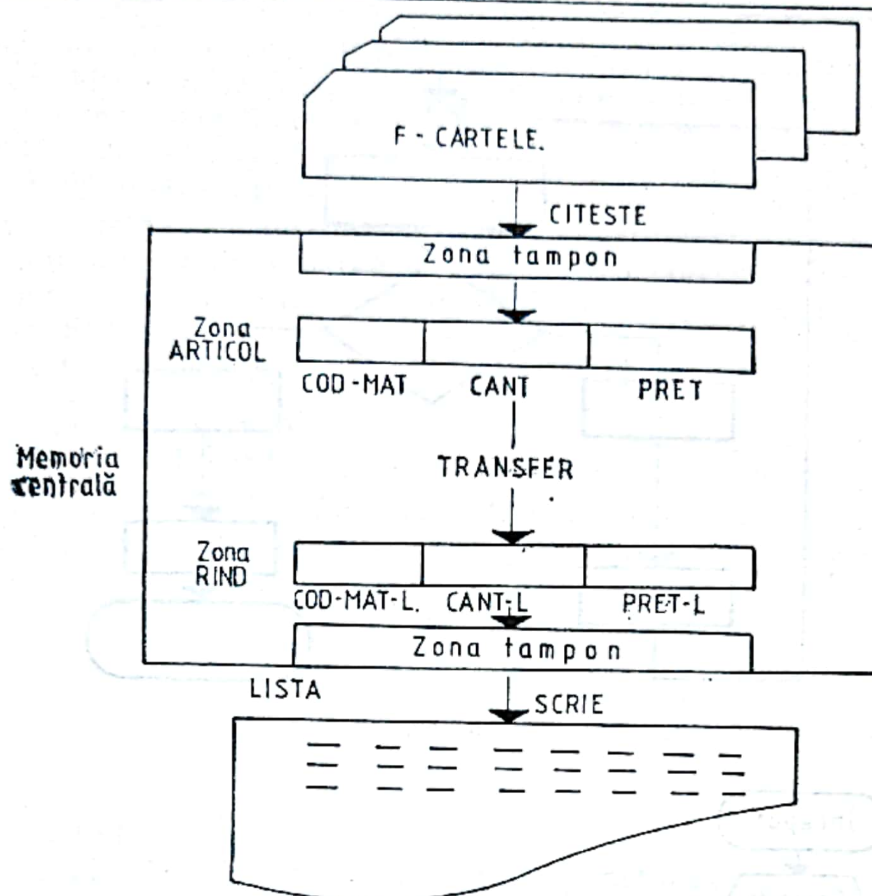

F - CARTELE.
CITESTE
Zona tampon
Zona ARTICOL
COD-MAT
CANT
PRET
Memoria centrală
TRANSFER
Zona RIND
COD-MAT-L.
CANT-L
PRET-L
Zona tampon
LISTA
SCRIE

Fig. 6.7.

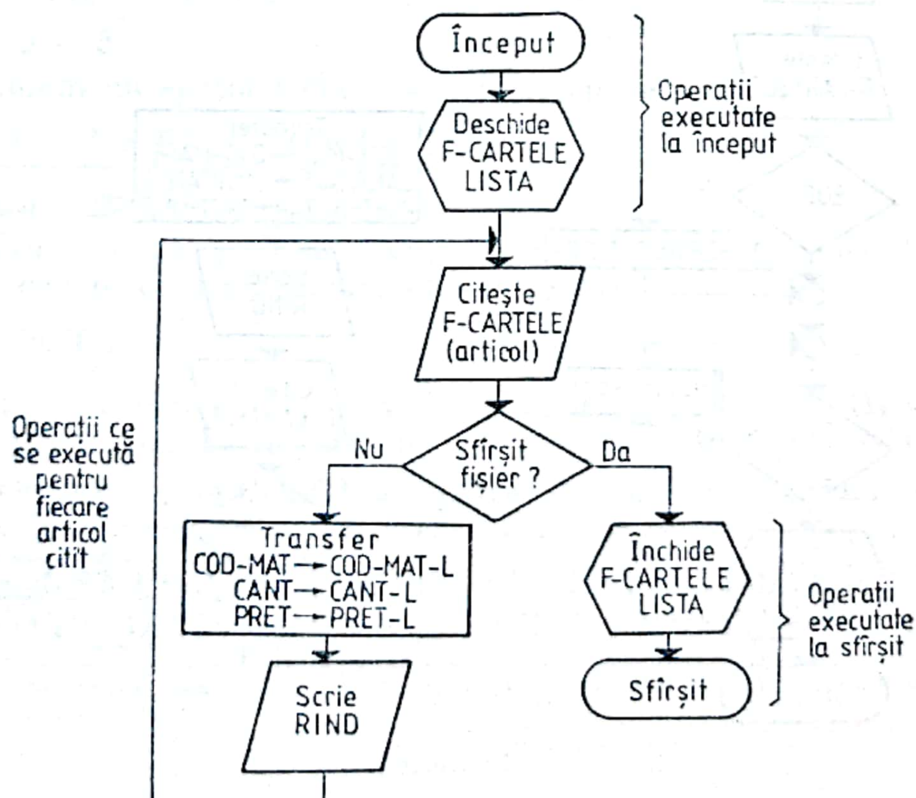

Început
Deschide F-CARTELE LISTA
Operații executate la început
Citește F-CARTELE (articol)
Operații ce se execută pentru fiecare articol citit
Nu
Sfîrșit fișier ?
Da
Transfer
COD-MAT→COD-MAT-L
CANT→CANT-L
PRET→PRET-L
Scrie RIND
Închide F-CARTELE LISTA
Sfîrșit
Operații executate la sfîrșit

Fig. 6.8.

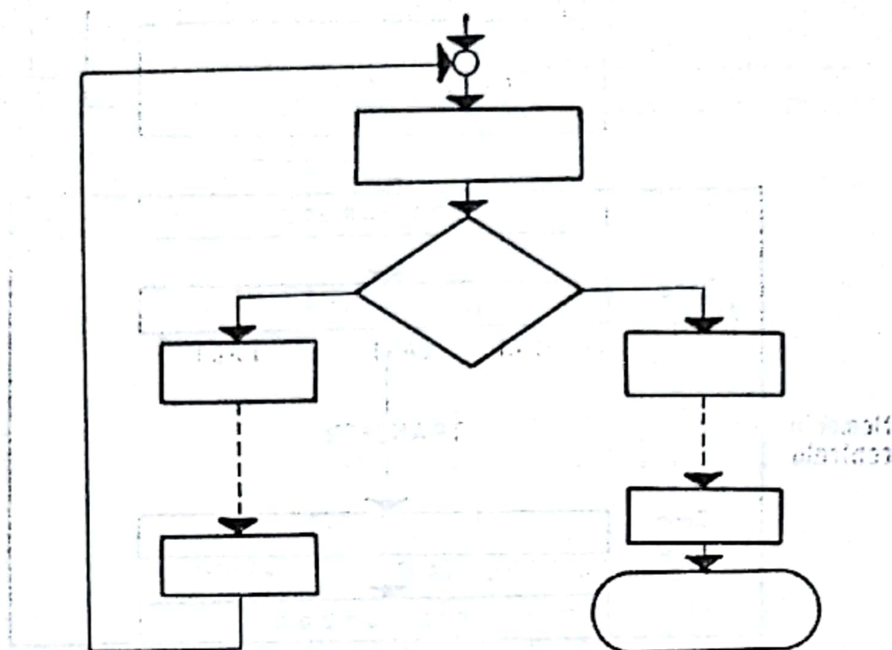


Fig. 6.9.

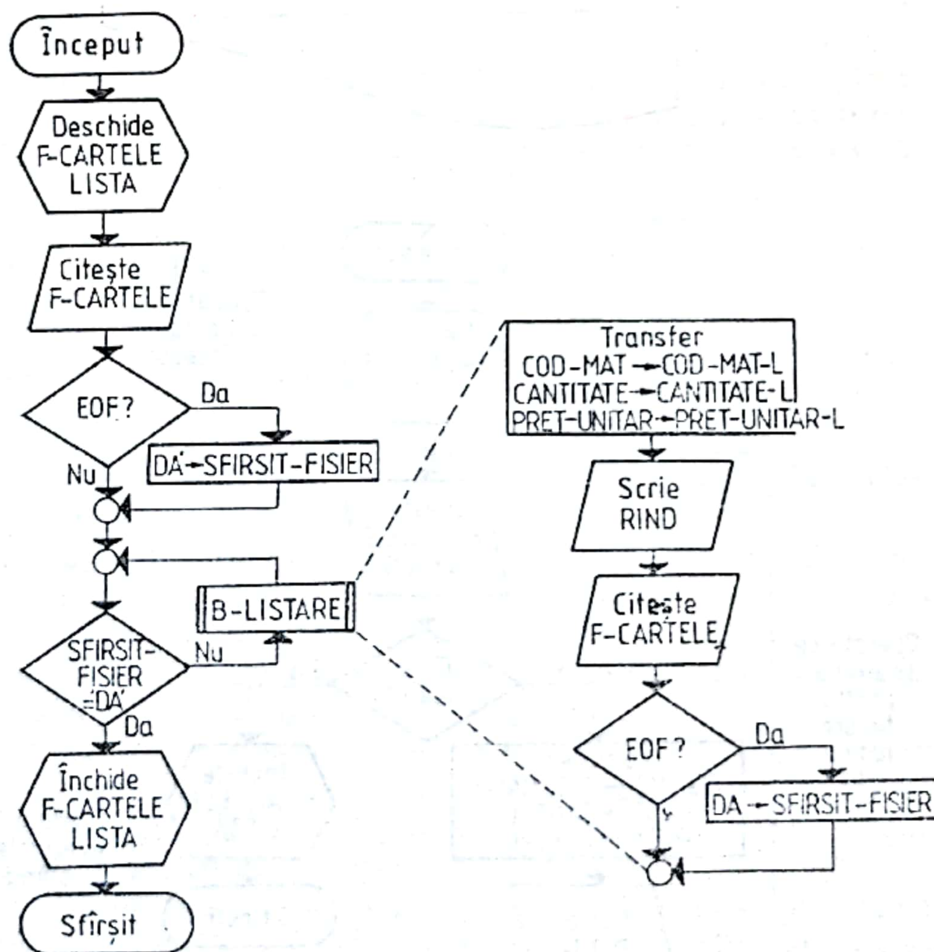


Fig. 6.10.

în zona articol din memorie. Pentru a fi înregistrat pe un alt suport (în cazul nostru hîrtia de la imprimantă), articolul trebuie transferat în zona articol corespunzătoare fișierului în ieșire (numită RÎND). Fiecărui fișier i se asociază deci cîte o zonă articol. Structura acestei zone trebuie descrisă de către programator prin program.

Folosind datele și schema logică de programare prezentate mai înainte, scrieți pe formularul de programare, programul corespunzător. Verificați răspunsul cu programul 6.1, din pag. 108 (în loc de SFIRȘIT-FIȘIER folosiți IND-SF).

Programul este asemănător?

DA → paragraful 6.28
NU ↺ paragraful 6.1.

6.28 Pentru ca un limbaj să fie înțeles trebuie ca la regulile de grupare a caracterelor și cuvintelor să se adauge și o anumită punctuație.

Sînt definite 7 (șapte) semne de punctuație:

.
,
;
(
)
'
ь — spațiul

6.29 Iată regulile de folosire a semnelor de punctuație;

Regula — 1

Orice frază* COBOL se termină cu un punct.

Regula — 2

Minimum un spațiu trebuie să figureze între două cuvinte.

|A|S|A| | |E|S|T|E| |C|O|R|E|C|T|

Regula — 3

Punctul, virgula, punctul și virgula, cînd sînt utilizate nu trebuie niciodată precedate ci doar urmate de un spațiu.

|P|U|N|C|T|.|

Utilizarea caracterelor , ; ca separatori este facultativă. Dacă ele apar în program trebuie să fie urmate în mod obligatoriu de spațiu.

6.30 *Î.* Orice program COBOL are patru părți numite diviziuni și anume:

. . . .

R. IDENTIFICATION DIVISION; ENVIRONMENT DIVISION; DATA DIVISION; PROCEDURE DIVISION

* O frază conține una sau mai multe instrucțiuni COBOL și se termină cu punct urmat de spațiu.

```
ID  DIVISION.
*************
PROGRAM-ID.
    SCOALA.
AUTHOR.

DATE-WRITTEN.
    20 DECEMBRIE 1980.
ENVIRONMENT DIVISION.
*********************
CONFIGURATION SECTION.
SOURCE-COMPUTER.
    FELIX C-256 CONFIGURATIE A-S-E.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT F-CARTELE ASSIGN  SYSIN.
    SELECT LISTA     ASSIGN  SYSOUT.
DATA DIVISION.
**************
FILE SECTION.
FD  F-CARTELE     RECORDING F
                  LABEL RECORD OMITTED.
01  ARTICOL.
  02 COD-MAT            PIC  9(7).
  02 CANTITATE          PIC 9(5)V99.
  02 PRET-UNITAR        PIC 999V99.
FD  LISTA         RECORDING F
                  LABEL RECORD OMITTED.
01  RIND.
  02 FILLER               PIC X.
  02   COD-MAT-L          PIC 9(7).
  02   CANTITATE-L        PIC BBZZZZ9.99.
  02   FILLER             PIC XX.
  02   PRET-UNITAR-L      PIC  ZZ9.99.
WORKING-STORAGE SECTION.
77  IND-SF                PIC XX VALUE 'NU'.
  88 IND-SF-DA                   VALUE 'DA'.
PROCEDURE DIVISION.
*******************
MODUL-PRINCIPAL.
    OPEN INPUT  F-CARTELE
         OUTPUT LISTA
    READ F-CARTELE AT END MOVE 'DA' TO IND-SF.
    PERFORM B-LISTARE UNTIL IND-SF-DA
    CLOSE F-CARTELE
          LISTA
    STOP RUN.
B-LISTARE.
**********
    MOVE SPACES            TO RIND
    MOVE COD-MAT           TO COD-MAT-L
    MOVE CANTITATE         TO CANTITATE-L
    MOVE PRET-UNITAR       TO PRET-UNITAR-L
    WRITE RIND AFTER 1
    READ F-CARTELE AT END MOVE 'DA' TO IND-SF.
                        PROGRAMUL 6.1
```

Programul 6.1.

6.31 Î. Fiecare diviziune poate conține secțiuni și paragrafe cu excepția . . . . care conține numai . . . .

R. diviziunii IDENTIFICATION; paragrafe

6.32 Î. În diviziunea de echipament se furnizează informații în legătură cu . . . .

R. configurația folosită, caracteristicile fișierelor și modul de tratare

---

6.33 Î. În diviziunea de date se descriu . . . .

R. fișierele și datele utilizate în prelucrare

6.34 Î. Diviziunea de procedură indică . . . .
și ordinea de executare a acestora. Printre instrucțiunile acestei diviziuni cunoașteți deja instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . . instrucțiunea . . . .
care realizează . . . .

R. operațiile de prelucrare
OPEN; deschiderea fișierelor
CLOSE; închiderea fișierelor
MOVE; un transfer (operație de atribuire)
READ; citirea unui articol (aducerea lui în memoria centrală)
WRITE; scrierea unui articol (a unui rînd la imprimantă)
GO TO; realizarea unui salt
PERFORM ... UNTIL; saltul cu revenire (specific structurii repetitive).

LECȚIA a 7-a INSTRUCȚIUNI DE CALCUL

- expresii aritmetice
- instrucțiunea ADD
- instrucțiunea MULTIPLY
- instrucțiunea SUBTRACT
- instrucțiunea DIVIDE
- instrucțiunea COMPUTE

7.1 Pentru efectuarea calculelor fiecare utilizator definește anumite formule sau expresii.

Spre exemplu pentru calcularea normativului de mijloace circulante la producția neterminată (la nivelul cheltuielilor de producție aferente fiecărui produs cuprins în plan) se folosește formula:

$$NV = \frac{CU}{360} \times D \times K$$

unde:

NV, normativul mediu anual pentru producția neterminată pe produs;
CU, cheltuielile anuale pe produs la cost de uzină;
D, durata ciclului de fabricație;
K, coeficientul de corectare specific fiecărui produs.

7.2 Limbajul COBOL permite obținerea unor expresii aritmetice, care să descrie astfel de formule, combinînd următoarele elemente:

— *operanzii*, care sînt date elementare numerice sau literali numerici;

— *operatorii aritmetici*, care indică operațiile de efectuat asupra datelor (operanzilor):

+ pentru adunare
— pentru scădere
* pentru înmulțire
/ pentru împărțire
** pentru ridicarea la putere

Exemple:

a) cîmpuri elementare numerice → literal numeric

NV = CU / 360 * D * K

→ operatori aritmetici

b) (A + B / K + D ** 3) * F / N

7.3 *Î.* Operaţia de ridicare la putere se precizează cu ajutorul operatorului numeric :

1 —
2 *
3 **

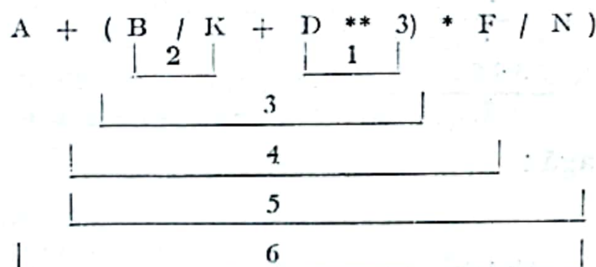
R. 3

7.4 Ordinea efectuării operaţiilor, dacă nu este specificată cu ajutorul parantezelor, este următoarea :

— operaţii de ridicare la putere;
— operaţii de împărţire şi înmulţire;
— operaţii de adunare şi scădere.

Observaţie

În condiţiile folosirii parantezelor, evaluarea expresiei începe cu parantezele interioare:



R1 = D ** 3
R2 = B / K
R3 = R1 + R2
R4 = R3 * F
R5 = R4 / N
R6 = R5 + A

unde :

R1—R5, reprezintă valorile rezultatelor intermediare obţinute în evaluarea expresiei aritmetice de mai sus

R6, valoarea rezultatului final.

7.5 *Î.* Fie expresia :

A + B — (C * D) / F

indicaţi ordinea de efectuare a operaţiilor

R. R1 = C * D
R2 = R1 / F
R3 = A + B
R4 = R3 — R2

7.6 La scrierea expresiilor operatorii trebuie să fie precedaţi şi urmaţi de spaţiu. Paranteza deschisă nu poate fi urmată de spaţiu dar trebuie precedată de cel puţin un spaţiu. Paranteza închisă nu poate fi precedată de spaţiu dar trebuie urmată de cel puţin un spaţiu.

Exemplu eronat:

| (| |A| | = | |B| | + | |C| – |D| |) |

Exemplu corect:

| | (|A| | = | |B| | + | |C| | – | |D|) | |

7.7 Rezultatul obţinut prin operaţiile de calcul se depune într-un cîmp receptor descris de către programator cu o anumită lungime. În cazul în care lungimea acestuia este mai mică decît valoarea rezultatului se face trunchierea.

7.8 Se poate ajunge la trunchierea rezultatului la partea zecimală:

Exemplu:

operand 1	|3|1|4|2|8|
+	↑
operand 2	|2|5|3|4|6|
	↑
rezultat	|3|3|9|6|2|
	↑

sau la partea întreagă:

Exemplu:

operand 1	|3|9|4|3|5|
+	
operand 2	|4|3|1|4|
rezultat	|3|7|4|9|

7.9 În anumite condiţii programatorul poate cere rotunjirea rezultatului.

Exemplu:

operand 1	|3|2|4|5|8|	|3|2|4|5|8|
+	↑	↑
operand 2	|1|7|3|4|6|	|1|7|3|4|8|
	↑	↑
rezultat	|3|4|1|9|2|	|3|4|1|9|3|
	↑ trunchiere	↑ rotunjire

Prin urmare la ultima zecimală a rezultatului se adaugă o unitate dacă zecimala care urmează este > 5.

7.10 În programare, prezintă interes şi semnalarea cazurilor de depăşire de capacitate, pentru a nu se denatura rezultatele calculelor.

7.11 Programatorul COBOL poate cere:

— *rotunjirea cifrelor din dreapta numărului* folosind în cadrul instrucţiunilor aritmetice clauza ROUNDED scrisă alături de cîmpul receptor a rezultatului de rotunjit

— *controlul depăşirii de capacitate* folosind clauza ON *SIZE ERROR* *.

* Şi împărţirea cu zero este asimilată unei depăşiri de capacitate.

Programatorul va menţiona modul de continuare a procesului de prelucrare în cazul în care la efectuarea unei instrucţiuni de calcul se ajunge la depăşirea capacităţii (după clauza ON *SIZE ERROR* sînt menţionate instrucţiunile de executat cînd se detectează depăşirea de capacitate).

Exemplu:

```
ADD VALOARE TO TOTAL
        ON SIZE ERROR
                MOVE 1 TO V-DEPAŞIRE-C.
```

Se adună conţinutul cîmpului VALOARE la cel al cîmpului TOTAL; dacă se înregistrează depăşire de capacitate se execută instrucţiunea MOVE.

Apariţia clauzei ON *SIZE ERROR* într-o instrucţiune aritmetică face ca aceasta să devină condiţională; în conceptul programării structurate instrucţiunii i se va asocia o variabilă ce poate lua valorile:

v_1 — valoare dinainte de efectuarea instrucţiunii aritmetice (în general 0);
v_2 — valoare luată dacă se constată o depăşire de capacitate (în general 1).

Exemplu:

```
MOVE 0 TO VARIABILA
COMPUTE A = B * C ON SIZE ERROR MOVE 1 TO VARIABILA.
*   IF     VARIABILA = 1 THEN
                    SECVENŢA CE CORESPUNDE
*                   CAZULUI DE DEPĂŞIRE A CAPACITĂŢII
*                           ELSE
*                   SECVENŢA CE CORESPUNDE
*                   CAZULUI ABSENŢEI DEPĂŞIRII
*                   DE CAPACITATE
```

Compilatorul COBOL, generează la întîlnirea acestei clauze o secvenţă de instrucţiuni cu ajutorul cărora sînt detectate cazurile de depăşire a capacităţii.

7.12 *Î.* Fie cîmpurile CANT, PRET şi VALOARE cu lungimile precizate în fig. 7.1.

Menţionaţi rezultatul real, rezultatul fără şi cu utilizarea clauzelor:

— ROUNDED
— ON SIZE ERROR

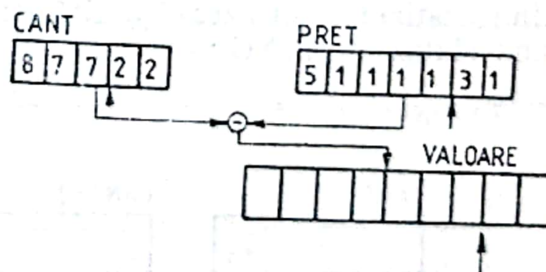


Fig. 7.1.

R. 44834883,3582
4834883,35
4834883,36

7.13 *Instrucţiunea ADD* se utilizează pentru cumulări de date

Exemple: fig. 7.2., fig. 7.3.

ADD VALOARE TO TOTAL—VALOARE, adună conţinutul cîmpului VALOARE la cel al cîmpului TOTAL—VALOARE.

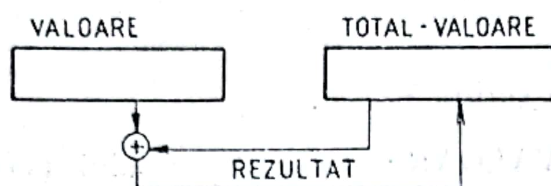


Fig. 7.2.

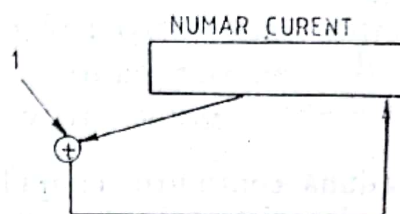


Fig. 7.3.

• *Formatul 1* al instrucţiunii ADD este:

ADD {nume—dată—1 | literal—1} [{nume—dată—2 | literal—2}] ...

TO nume—dată—n [nume—dată—m] ...

[**ROUNDED**]
[ON **SIZE ERROR** instrucţiune ....]

iar logica derulării în calculator cea din fig. 7.4.

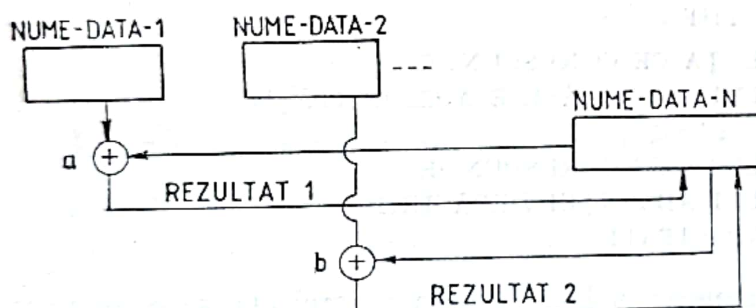


Fig. 7.4.

Observaţie. Cînd se realizează adunarea prin cumulare la conţinutul unui cîmp, programatorul trebuie să asigure o valoare iniţială pentru cîmpul receptor (în principiu aceasta este zero).

7.14 Pentru a efectua operaţii de adunare cu depunerea rezultatului într-un cîmp distinct se utilizează o altă structură a instrucţiunii de adunare, folosind clauza GIVING:

Exemplu: fig. 7.5.

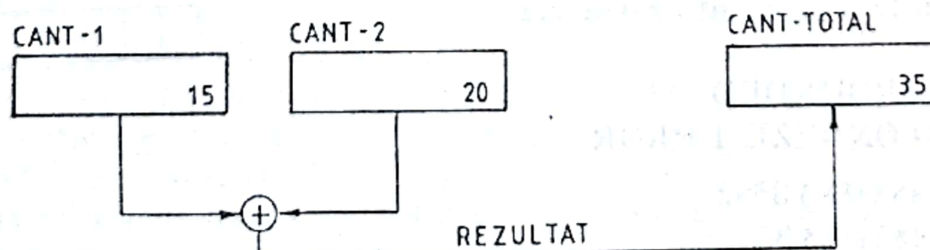


Fig. 7.5.

ADD CANT—1 CANT—2 GIVING CANT—TOTAL

• *Formatul—2* al instrucţiunii ADD este :

ADD {nume—dată—1 | literal—1} [{nume—dată—2 | literal—2}]...
  GIVING nume—dată—n
  [**ROUNDED**]
  [ON **SIZE ERROR** instrucţiunea ...]

Logica execuţiei instrucţiunii ADD este :

— *în formatul 1*, la valoarea fiecărei date *nume dată—n, nume—dată—m...* ... se adună valorile datelor *nume—dată—1/literal—1 nume—dată—2/literal 2* ...

— *în formatul 2*, se însumează valorile datelor *nume—dată—1/literal—1, nume—dată—2/literal—2* iar rezultatul se atribuie datei *nume—dată—n*

7.15 *Î.* Care este diferenţa dintre instrucţiunea de adunare în formatul 1 şi formatul 2?

R. În formatul 1 datele se cumulează la conţinutul cîmpului receptor iar în formatul 2 datele se însumează iar rezultatul se depune în cîmpul receptor.

7.16 *Î.* Scrieţi instrucţiunea de adunare pentru expresiile :
TOTAL := VALOARE—1 + VALOARE—2 + VALOARE—3
TOTAL := TOTAL + VALOARE

R. ADD VALOARE—1 VALOARE—2 VALOARE—3 GIVING TOTAL

ADD VALOARE TO TOTAL

Răspunsurile dumneavoastră sînt corecte?

DA → 7.17
NU ↺ 7.13.

7.17 *Instrucţiunea MULTIPLY* (fig. 7.6) se foloseşte pentru efectuarea operaţiilor de înmulţire ; în *varianta a* rezultatul înmulţirii se depune în cîmpul de memorie afectat celui de al doilea operand (anulînd deci valoarea acestuia), iar *în varianta b* rezultatul înmulţirii se depune într-un cîmp special rezervat (fig. 7.7).

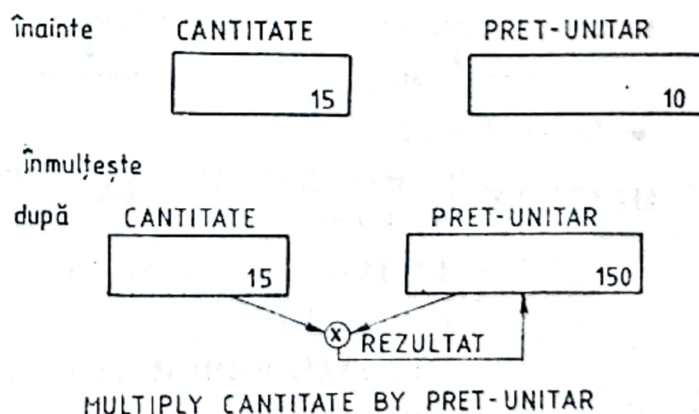
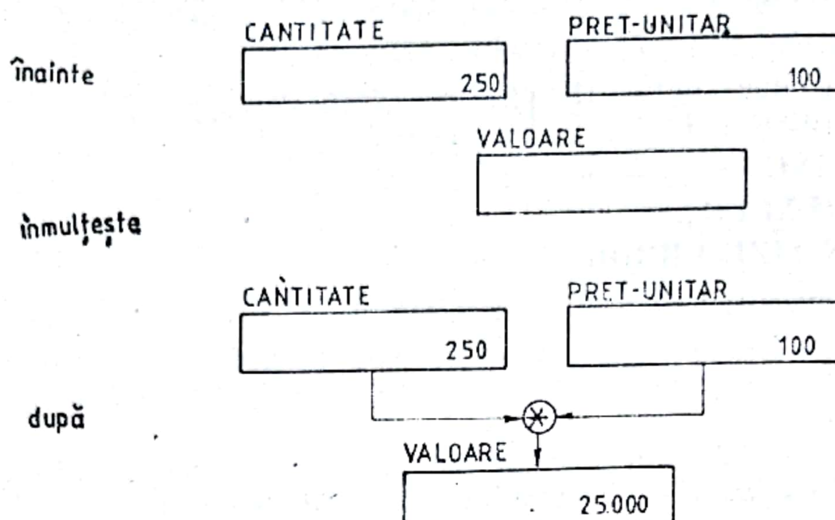


Fig. 7.6.



MULTIPLY CANTITATE BY PRET-UNITAR GIVING VALOARE

Fig. 7.7.

7.18 *Î.* Avantajul variantei b constă în faptul că . . . .

R. păstrează neschimbat conţinutul celui de al doilea factor

7.19 Instrucţiunea de înmulţire poate avea două formate:

- *formatul—1*:

MULTIPLY {nume—dată—1 | literal—1} **BY** nume—dată—2

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiunea . . . .]

Exemple:

MULTIPLY 360 BY VALOARE
(360 → literal—1; VALOARE → nume—dată—1)

MULTIPLY CÎMP—1 BY CÎMP—2
(CÎMP—1 → nume—dată—1; CÎMP—2 → nume—dată—2)

- *formatul —2*:

MULTIPLY {nume—dată—1 | literal—1} **BY** {nume—dată—2 | literal—2}

GIVING nume—dată—3

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiunea . . . .]

unde:

nume—dată—3, poate fi un cîmp numeric de calcul sau de editare.

Logica execuţiei instrucţiunii MULTIPLY este:

• *în formatul 1*, valoarea datei *nume—dată—1/literal—1* se înmulţeşte cu cea a datei *nume—dată—2* iar rezultatul se atribuie datei *nume—dată—2*;

• *în formatul—2*, valoarea datei *nume—dată—1/literal 1* se înmulţeşte cu cea a datei *nume—dată—2/literal—2* iar rezultatul se atribuie datei *nume—dată—3*.

Exemple:

```
MULTIPLY 545 BY VALOARE—1 GIVING VALOARE—2
MULTIPLY ZONA—1 BY ZONA—2 GIVING ZONA—3
```

7.20 *Î.* Iată un exerciţiu pentru a vedea dacă v-aţi însuşit instrucţiunea MULTIPLY

	MULTIPLY	BY	GIVING
1.	nume—dată—1	nume—dată—2	literal—1
2.	nume—dată—1	literal—1	
3.	nume—dată—1	nume—dată—2	
4.	literal—1	nume—dată—2	literal—2
5.	literal—1	nume—dată—1	nume—dată—2
6.	literal—1	literal—2	literal—3
7.	nume—dată—1	literal—1	literal—2
8.	literal—1	literal—2	

R. 1: incorect, 2: incorect, 3: corect, 4: incorect, 5: corect, 6: incorect, 7: incorect, 8: incorect

Răspunsurile sînt corecte?

DA → 7.21
NU ↶ 7.17

7.21 *Instrucţiunea SUBTRACT* se foloseşte pentru efectuarea operaţiilor de scădere:

• *starea iniţială:*

A: 10 B: 103

scade A din B

SUBTRACT A FROM B

• *starea finală:*

A: 10 B: 93

După efectuarea scăderii cîmpul B conţine valoarea rezultatului. În cazul în care dorim să utilizăm ulterior conţinutul cîmpurilor A şi B, de dinaintea scăderii, atunci comanda de scădere se va da în felul următor (fig. 7.8).

7.22 *Î.* Precizați rezultatele operației de scădere în condițiile în care cîmpurile implicate A, C, B, și V au valorile:

A = 100 B = 250 C = 300 V = 998 U:

a) — după executarea instrucțiunii de scădere

SUBTRACT A B C FROM V

. . . .
. . . .
. . . .

b) — dar după executarea instrucțiunii de scădere

SUBTRACT A B C FROM V GIVIN G U

stare inițială

A 10 B 103

C

SUBTRACT A FROM B GIVING C

din depune rezultatul în

stare finală

A 10 B 103

C 93

Fig. 7.8.

R. a) A = 100; B = 250; C = 300; V = 348

b) A = 100; B = 250; C = 300; V = 998; U = 348

7.23 Instrucțiunea SUBTRACT poate avea mai multe formate de utilizare:

- *formatul—1*

SUBTRACT {nume—dată—1 | literal—1} [{nume—dată—2 | literal—2}] . . . **FROM** nume—dată—n

[**ROUNDED**]

[ON **SIZE ERROR** instrucțiunea . . . .]

Exemple:

```
SUBTRACT 565 FROM ZONA-1
SUBTRACT ZONA FROM ZONA-1
SUBTRACT A B C FROM D
```

- *formatul—2*

SUBTRACT {nume—dată—1 | literal—1} [{nume—dată—2 | literal—2}] . . . **FROM** {nume—dată—m | literal—m}

GIVING nume—dată—n

[**ROUNDED**]

[ON **SIZE ERROR** instrucțiune . . . .]

Exemple:

```
SUBTRACT ZONA-1 ZONA-2 ZONA-3 GIVING ZONA-4
SUBTRACT 153 FROM ZONA GIVING REZULTAT
```

Cîmpul receptor în cazul folosirii opţiunii GIVING poate fi descris cu şablon pentru cîmpuri de calcul sau de editare.

Logica execuţiei instrucţiunii SUBTRACT este:

- *în formatul—1*, valorile datelor *nume—dată—1/literal—1*, *nume—dată—2/literal—2* ... sînt însumate; rezultatul însumării se scade din valoarea datei *nume—dată—n*, iar rezultatul scăderii se atribuie datei *nume—dată—n*;
- *în formatul—2*, valorile datelor *nume—dată—1/literal—1*, *nume—dată—2/literal—2* ... sînt însumate; rezultatul însumării se scade din valoarea datei *nume—dată—m/literal—m* iar rezultatul scăderii se atribuie datei *nume—dată—n*.

7.24 *Instrucţiunea DIVIDE* se foloseşte pentru efectuarea operaţiilor de împărţire si are mai multe formate.

Exemplu simplificat de utilizare:

starea iniţială:

A	B
100	5

operaţie de împărţire A: B
DIVIDE B INTO A

starea finală:

A	B
20	5

Rezultatul este atribuit cîmpului A care se află în cadrul instrucţiunii după cuvîntul INTO. Pentru a păstra conţinutul cîmpului A de dinaintea împărţirii se utilizează clauza GIVING.

starea iniţială: (cîmpul C serveşte pentru depunerea rezultatelor)

A	B	C
100	5	

DIVIDE B INTO A GIVING C

starea finală:

A	B	C
100	5	20

Cîmpul receptor poate fi de natură numerică (de calcul sau de editare).

7.25 Instrucţiunea DIVIDE prezintă mai multe formate:

- *formatul—1*

```
DIVIDE  {nume—dată—1}   INTO  nume—dată—2
        {literal—1  }
        [ROUNDED]
        [ON SIZE ERROR  instrucţiunea....]
```

Exemple:

DIVIDE A INTO C pentru C / A
DIVIDE 5 INTO B pentru B / 5

- *formatul—2*

```
DIVIDE  {nume—dată—1}   INTO  {nume—dată—2}
        {literal—1  }         {literal—2  }
        GIVING     nume—dată—3
        [REMAINDER nume—dată—4]
        [ROUNDED]
        [ON SIZE ERROR  instrucţiunea ....]
```

Exemple:

```
DIVIDE B INTO C GIVING D D ←C/B
DIVIDE B INTO C GIVING CIT REMAINDER REST
```

- *formatul—3*

```
DIVIDE  {nume—dată—1}   BY  {nume—dată—2}
        {literal—1  }       {literal—2  }
        GIVING  nume—dată—3
        [REMAINDER  nume—dată—4]
        [ROUNDED]
        [ON SIZE ERROR  instrucţiune ....]
```

Exemplu:

```
DIVIDE B BY C GIVING CIT REMAINDER REST
```

unde:

- CIT, numele cîmpului folosit pentru memorarea cîtului împărţirii;
- REST, numele cîmpului folosit pentru memorarea restului împărţirii.

Clauza REMAINDER se foloseşte pentru a atribui unui cîmp valoarea restului.

Logica execuţiei instrucţiunii DIVIDE este:

- *în formatul—1*, valoarea datei *nume—dată—2* se împarte la valoarea datei *nume—dată—1/literal—1* iar cîtul se atribuie datei *nume—dată—2*;
- *în formatul—2* valoarea datei *nume—dată—2* se împarte la valoarea datei *nume—dată—1/literal—1*, cîtul se atribuie datei *nume—dată—3*, iar restul datei *nume—dată—4* (afirmaţia pentru rest este valabilă numai în cazul folosirii clauzei REMAINDER);
- *în formatul—3* valoarea datei *nume—dată—1/literal—1* se împarte la valoarea datei *nume—dată—2/literal—2*; valoarea cîtului se atribuie datei *nume—dată—3*, iar a restului datei *nume—dată—4*.

7.26 Paragrafele precedente au demonstrat că instrucţiunile ADD, MULTIPLY, SUBTRACT, DIVIDE se manipulează cu greutate, mai ales cînd expresia de calcul este mai complexă.

7.27 *Î.* Fie expresia de calcul:

$$\frac{A - (B + C)}{D} \times E$$

Descompuneţi expresia în operaţii simple de calcul utilizînd numai instrucţiunile ADD, SUBTRACT, MULTIPLY, DIVIDE

R. — REZULTAT—1 = B + C
ADD B C GIVING REZULTAT—1
— REZULTAT—2 = A — REZULTAT—1
SUBTRACT REZULTAT—1 FROM A GIVING REZULTAT—2
— REZULTAT—3 = REZULTAT—2/D
DIVIDE D INTO REZULTAT—2 GIVING REZULTAT—3
sau
DIVIDE REZULTAT—2 BY 3 GIVING REZULTAT—3
— REZULTAT—FINAL = REZULTAT—3 * E
MULTIPLY REZULTAT—3 BY E GIVING REZULTAT—FINAL

Răspunsul este corect
DA ↰ 7.13
NU → 7.28

7.28 Pentru expresii aritmetice se indică folosirea unei singure instrucţiuni de calcul numită COMPUTE.

Exemplu:

COMPUTE REZULTAT = A — (B + C) / D * F

Formatul general al instrucţiunii COMPUTE este

COMPUTE nume—dată—1

[**ROUNDED**] = {nume—dată—2 | literal | expresie—aritmetică}

[ON **SIZE ERROR** instrucţiune ....]

unde:

nume—dată—1 desemnează numele unui cîmp numeric sau de editare.

Exemple:

COMPUTE A = B
COMPUTE A = B + C — D
COMPUTE A = B + C(A + D / 100)

Cîmpului *nume—dată—1* i se atribuie prin instrucţiunea COMPUTE valoarea cîmpului *nume—dată—2/literal* sau valoarea expresiei aritmetice.

7.29 *Î.* Instrucţiunile ADD, MULTIPLY, DIVIDE, SUBTRACT se folosesc pentru efectuarea operaţiilor de: . . . .
iar COMPUTE pentru calcule . . . .

R. adunare, înmulţire, împărţire, scădere, complexe

7.30 *Î.* Scrieţi în tabelul de mai jos instrucţiunile pentru efectuarea calculelor:

Calcule de efectuat	instructiuni
NC : = NC + 1	
T—VAL : = T—VAL + VAL	
T—VALOARE : = VAL1 + VAL2 + VAL3	
ZD : = DM — AB + BC	
F : = A — IB + C/100	

	Calcule de efectuat	Instrucțiuni
R.	NC: = NC + 1	ADD 1 TO NC
	T—VAL: = T—VAL + VAL	ADD VAL TO T—VAL
	T—VALOARE = VAL1 + VAL2 + VAL3	ADD VAL1 VAL2 VAL3 GIVING T—VALOARE
	ZD = DM — AB + BC	COMPUTE ZD = DM — AD + BC
	F = A — IB + C/100	COMPUTE F = A — IB + C/ 100

7.31 *Î.* Clauza ROUNDED se folosește cînd . . . .
iar clauza ON SIZE ERROR . . . .

R. — se cere rotunjirea rezultatului

— cînd se cere, la apariția unei depășiri de capacitate, să se execute o anumită instrucțiune (sau un grup de instrucțiuni delimitat prin punctul terminal).

7.32 Limbajul COBOL permite folosirea în cadrul operațiilor aritmetice a unor operanzi cu reprezentări diferite. Pentru realizarea operației aritmetice este necesară conversia operanzilor la același format de reprezentare, care poate fi zecimal-împachetat, cod complementar sau virgulă mobilă și corespunde unui operator aritmetic al unității centrale.

7.33 Fie spre exemplu următoarea structură de articol asociat unui fișier memorat pe disc magnetic:

```
01 ARTICOL—INTRĂRI.
   02 GESTIUNE                PIC 99.
   02 COD—PRODUS              PIC 9(5).
   02 COST—NORMAT             PIC 9(4)V99.
   . . . . . . . . . . . . . . . . . . . .
```

Se cere însumarea valorilor luate de cîmpul COST—NORMAT pentru furnizarea totalurilor la nivel produs (pentru același produs în fișierul pe disc magnetic există mai multe articole) și gesiune.

Pentru realizarea însumărilor în W—S se descriu cîmpurile:
TOTAL—COST—NORMAT—PRODUS și
TOTAL—COST—NORMAT—GESTIUNE.

7.34
```
WORKING—STORAGE SECTION.
01 DATE—INTERMEDIARE.
   02 TOTAL—COST—NORMAT—PRODUS    PIC 9(6)V99
                                   VALUE 0.
   02 TOTAL—COST—NORMAT—GESTIUNE PIC 9(8)V99
                                   VALUE 0.
01 RÎND—TOTAL.
   02 TOTAL—IMPRIMANTA             Z(7)9.99.
```

*) W—S de la WORKING—STORAGE

7.35 Pentru efectuarea calculelor și pregătirea imaginii articolelor de editat se folosesc instrucțiunile ADD și MOVE în următoarea structură:

```
ADD COST—NORMAT TO TOTAL—COST—NORMAT—PRODUS
MOVE TOTAL—COST—NORMAT—PRODUS TO TOTAL—
                                         IMPRIMANTA
ADD TOTAL—COST—NORMAT—PRODUS TO
                        TOTAL—COST—NORMAT—GESTIUNE
MOVE TOTAL—COST—NORMAT—GESTIUNE TO
                              TOTAL—IMPRIMANTA
```

7.36 Realizarea grupului de instrucțiuni ADD și MOVE implică următoarele operații:

— conversia valorii cîmpului *operand—1* în zecimal—împachetat (în cazul de mai sus a cîmpurilor COST—NORMAT și respectiv TOTAL—COST—NORMAT—PRODUS);

— conversia valorii cîmpului *operand—2* în zecimal—împachetat (în cazul de mai sus a cîmpurilor TOTAL—COST—NORMAT—PRODUS și respectiv TOTAL—COST—NORMAT—GESTIUNE);

— însumarea celor două valori;

— conversia valorii rezultate din calcul (adunare) în zecimal—despachetat și memorarea în zona desemnată prin operand—2;

— transferul rezultatului în cîmpul TOTAL—IMPRIMANTA în vederea editării.

7.37 Pentru a optimiza numărul de conversii, se va folosi obțiunea COM—3 care indică memorarea datelor în zecimal împachetat (vezi lecția 10).

```
01 . ARTICOL—INTRĂRI.
   02  GESTIUNE              PIC 99.
   02  COD—PRODUS            PIC 9(5).
   02  COST—NORMAT           PIC 9(4)V99 COMP—3.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   WORKING—STORAGE—SECTION.
01 DATE—INTERMEDIARE.
   02  TOTAL—COST—NORMAT—PRODUS PIC 9(6)V99
                                     VALUE 0
                                     COMP—3.
   02  TOTAL—COST—NORMAT—GESTIUNE PIC 9(8)V99
                                       VALUE 0
                                       COMP—3.
   :
01 RÎND—TOTAL.
   :
   02  TOTAL—IMPRIMANTA      PIC Z(7)9.99.
```

7.38 În varianta prezentată la 7.37 are loc conversia datelor memorate în cîmpurile TOTAL—COST—NORMAT—PRODUS și TOTAL—COST—NORMAT—GESTIUNE în zecimal despachetat și memorarea acestora în zona TOTAL—IMPRIMANTA (conversia implicată de forma diferită de reprezentare a datelor din instrucțiunea MOVE).

LECŢIA a 8-a TESTAREA CONDIŢIILOR ŞI INSTRUCŢIUNEA IF. VALIDAREA DATELOR

- testarea condiţiilor
  - testul de relaţie
  - testul de natură
  - testul de semn
  - testul nume de condiţie
- instrucţiunea IF
- clauza REDEFINES
- instrucţiunea EXAMINE

8.1 Operaţiile de testare a condiţiilor se realizează în limbajul COBOL folosind instrucţiunile:

```
IF
PERFORM . . . . . .UNTIL . . . . . .
PERFORM . . . . . . VARYING . . . . . .
SEARCH
```

8.2 Aşa cum s-a prezentat în lecţia 3 formula generală pentru realizarea operaţiei de testare este:

IF condiţie

THEN acţiunea—1
[**ELSE** acţiunea—2].

unde:

— *acţiunea—1/acţiunea—2* cuprind una sau mai multe instrucţiuni COBOL (inclusiv instrucţiunea de realizare a testării).

Valoarea logică a unei condiţii poate fi *„adevărat"* şi se execută *acţiunea—1* sau *„fals"* şi se execută *acţiunea—2*.

8.3 Limbajul COBOL admite două categorii de condiţii:

— *simple* care permit realizarea operaţiilor de testare în vederea determinării:

— relaţiei dintre doi termeni (test de relaţie);
— categoriei unei date (test de natură sau clasă);
— valorii algebrice a unei date (test de semn);
— egalităţii între valorile a două date sau apartenenţa la o anumită mulţime de valori (test nume de condiţie).

— *compuse* formate cu ajutorul condiţiilor simple, operatorilor logici AND, OR şi NOT şi, dacă este cazul, al parantezelor.

8.4 În exprimarea condiţiilor se utilizează:

— *operatorii relaţionali*:

> sau GREATER THAN

= sau EQUAL TO
< sau LESS THAN

— *operatorii logici*:
AND — și
OR — sau
NOT — negația (pentru a suplini lipsa operatorilor $\geqslant$, $\leqslant$, $\neq$)
— *operatorii aritmetici*
— *parantezele*
— *operanzii*.

Observație. Parantezele sînt utilizate în cazul formulării unei condiții compuse, și impun respectarea regulilor precizate la evaluarea expresiilor aritmetice.
În cazul în care într-o condiție compusă nu se folosesc parantezele ierarhia operatorilor este: operatorii aritmetici, operatorii de relație și operatorii logici (NOT, AND, OR).

8.5 *Testul de relație* se utilizează pentru a stabili dacă valoarea unei *dată/literal/expresie aritmetică* este mai mare, mai mică sau egală cu valoarea unei alte *dată/literal, expresie aritmetică*.

Condiția testată pentru a determina relația dintre doi termeni are formatul:

```
{nume—dată—1          }          [ { >              } ]  {nume—dată—2          }
{literal—1            }  IS [NOT] [ {GREATER THAN    } ]  {literal—2            }
{expresie—aritmetică  }          [ { =              } ]  {expresie—aritmetică  }
                                  [ {EQUAL TO        } ]
                                  [ { <              } ]
                                  [ {LESS THAN       } ]
```

Exemple:

```
IF STOC—EFECTIV > STOC—NORMAT ......
IF STOC—EFECTIV GREATER THAN STOC—NORMAT ......
IF A + 550 < B ......
```

Dacă cei doi termeni se află în relația precizată de către operatorul de relație, valoarea logică a expresiei este „*adevărat*" iar în caz contrar „*fals*". Folosirea operatorului logic NOT are ca efect inversarea celor două valori logice. Așa cum am menționat operatorul logic NOT suplinește lipsa operatorilor de relație pentru cazurile $\neq$, $\geqslant$, $\leqslant$.

La realizarea testului de relație trebuie avută în vedere natura cîmpurilor. În cazul în care cei doi termeni sînt numerici se compară valorile acestora iar dacă sînt nenumerici se compară începînd din partea stîngă, caracter cu caracter (pentru termenul de lungime mai mică se asigură completarea automată, cu caractere *spații*, pînă cînd și acest termen este adus la aceeași lungime). Dacă unul din termenii condiției are reprezentare zecimalîmpachetată, virgulă mobilă, sau binară operația de testare nu este precedată de operația de conversie (în vederea aducerii la aceeași formă de reprezentare în memorie).

8.6 *Î*. Inlocuiți termenii indicați în tabelul 8.1.

8.7 *Testul de natură a unei date* se utilizează pentru a stabili dacă valoarea unui cîmp este de natură numerică sau alfanumerică.

Condiția testată are formatul general:

{nume—dată / expresie—aritmetică} IS [NOT] {NUMERIC / ALPHABETIC}

unde:

nume — dată poate desemna: — în cazul opțiunii NUMERIC un cîmp numeric (reprezentat în zecimal împachetat sau în zecimal neîmpachetat) alfanumeric, sau un cîmp care conține numai caractere numerice:

— *în cazul opțiunii ALPHABETIC*, un cîmp nenumeric sau un cîmp grup (care conține caractere alfabetice și caracterul b)

Tabelul 8.1

A+B > C	
A < F	
Z = P+D	
	N GREATER THAN M
	B LESS THAN F

R.

A+B > C	A + B GREATER C
A < F	A LESS F
Z = P+D	Z EQUAL TO P+D
N > M	N GREATER THAN M
B < F	B LESS THAN F

Exemple:

IF DENUMIRE—PRODUS IS ALPHABETIC ......
IF PRET—UNITAR IS NOT NUMERIC ......

8.8 *Testul de semn* se utilizează pentru a stabili dacă valoarea unui cîmp este mai mică, mai mare sau egală cu zero.

Condiția testată are formatul general:

{nume—dată / expresie—aritmetică} IS [NOT] {POZITIVE / NEGATIVE / ZERO}

Exemple:

```
IF SOLD       IS POZITIVE ......
IF DIFERENTA IS NOT ZERO ......
IF DIFERENTA IS NEGATIVE ......
```

8.9 *Testul nume de condiție* se utilizează pentru a vedea dacă valoarea unui cîmp este egală cu o anumită valoare sau dacă este cuprinsă într-un interval de valori.

Limbajul COBOL dă posibilitatea programatorului să asocieze unei valori sau unui interval de valori (după caz a unor intervale de valori) un *nume — dată* numit și nume—condiție.

Data *nume—condiție* se descrie cu ajutorul unei rubrici de nivel 88, ea trebuie să urmeze celei de descriere a datei testate.

Exemple:

```
02 STARE—CIVILĂ       PIC 9.
   88 CASATORIT VALUE 1.
   88 NECASATORIT VALUE 2.
.
.
.
.
```

```
02 COD-GESTIUNE      PIC 99.
   88 GEST-VAL-ADMISA VALUE 0 THRU 30.
.
.
.
```

Formatul general al rubricii de descriere a numelui de condiție este:

```
88 nume-condiție  {VALUE IS  } literal-1 {THROUGH} literal-2
                  {VALUES ARE}           {THRU   }

              [literal-3 [{THROUGH} literal-4]] ...
                          {THRU   }
```

unde:

— *literal—1* o valoare dată sau limita inferioară a unui interval de valori;
— *literal—2* limita superioară a intervalului de valori;
— *literal—3 literal—4* literale numerice și corespund limitelor inferioare și respectiv superioare în cazul utilizării unor intervale de valori.

Exemple de folosire a testului nume de condiție:

```
IF CASATORIT
          THEN PERFORM PROCEDURA-CASATORIT
. . .
IF GEST-VAL-ADMISA
          THEN
            PERFORM PROCEDURA-PRELUCRARE
          ELSE
            PERFORM PROCEDURA-GEST-VAL-ER.
```

8.10 *Condițiile compuse* se obțin folosind condițiile simple, operatorii logici și eventual parantezele.

Valorile logice ale condițiilor compuse în care se folosesc operatorii logici sînt următoarele:

C_1	C_2	C_1 AND C_2	C_1 OR C_2
a	a	a	a
a	f	f	a
f	a	f	a
f	f	f	f

unde: a — adevărat, f — fals

Exemple:

```
a) IF CANTITATE NUMERIC AND PRET-UNITAR NUMERIC
         THEN
           MULTIPLY CANTITATE BY PRET-UNITAR GIVING VALOARE
           ADD VALOARE TO TOTAL-VALOARE
         ELSE
           DISPLAY 'DATE NENUMERICE'
           ADD 1 TO C-ERORI
           MOVE 1 TO V.
b) IF NOT (GESTIUNE = 1 OR GESTIUNE = 2 OR GESTIUNE = 3)
         THEN
           PERFORM PROCEDURA-PRELUCRARE
         ELSE
           PROCEDURA-EROARE.
```

sau

```
IF NOT (GESTIUNE = 1 OR = 2 OR = 3) PERFORM PROCEDURA—PRE-
LUCRARE
        ELSE PERFORM PROCEDURA—EROARE.
```

Observație

Cele două instrucțiuni IF de la punctul b sînt echivalente

c)
```
IF COD—GESTIUNE > 0     AND COD—GESTIUNE < 15
        THEN
          PERFORM PROCEDURA—CALCUL
        ELSE
          PERFORM PROCEDURA—EROARE.
```

d)
```
IF X + 5 > Y AND Y < 1 500
        THEN
          PERFORM PROCEDURA—PRELUCRARE
        ELSE
          PERFORM PROCEDURA—EROARE.
```

8.11 Î. Condițiile pot fi . . . . sau . . . .

R. simple
compuse

8.12 Î. Condițiile simple se utilizează pentru a realiza:

. . . .
. . . .
. . . .
. . . .

R. testul de relație
testul de natură (clasă)
testul de semn
testul nume de condiție

8.13 Î. La formularea condițiilor compuse se utilizează:

. . . .
. . . .

R. condițiile simple
operatorii logici
parantezele

8.14 Î. Scrieți instrucțiunea IF și condiția compusă pentru a realiza operația de testare cunoscînd că se va apela procedura PROCEDURA—1 numai dacă cîmpul CANTITATE conține date numerice și cuprinse între 0 și 50 000

. . . .
. . . .

R.
```
IF CANTITATE NUMERIC AND (CANTITATE > 0 AND CAN-
TITATE < 50000)
   THEN
      PERFORM PROCEDURA—1.
```

8.15 Instrucţiunea IF, codifică structura alternativă şi precizează ordinea de execuţie a instrucţiunilor (procedurilor) în funcţie de valoarea logică obţinută prin testarea unei condiţii (simple sau compuse);

Format general:

```
IF condiţie
   THEN {instrucţiune—1;  [instrucţiune—2]} ...
        {NEXT SENTENCE                    }
 [ ELSE {instrucţiune—3;   [instrucţiune—4] ... ] .
        {NEXT SENTENCE                          ]
```

Principiul de execuţie al instrucţiunii IF este:

— se stabileşte valoarea logică a condiţiei (care poate fi „adevărat" sau „fals");

— dacă răspunsul este „adevărat" se execută instrucţiunea (instrucţiunile) prevăzută(e) după delimitatorul THEN;

— dacă răspunsul este „fals", se execută instrucţiunea (instrucţiunile) care apare după clauza ELSE;

— se realizează saltul la instrucţiunea de după punct.

Observaţie În condiţiile în care delimitatorul THEN este urmat de opţiunea NEXT SENTENCE se realizează de asemenea saltul la instrucţiunea de după punct.

8.16 **Exemple de** utilizare a instrucţiunii IF:

```
a)  IF C + D > E THEN ADD 1 TO V.
b)  IF A NOT = B  ADD VAL TO TOTAL—1
      ELSE        ADD VAL TO TOTAL—2.
c)  IF A < B NEXT SENTINCE
          ELSE MOVE DATA—1 TO DATA-2.
```

8.17 Secvenţa de instrucţiuni executată în funcţie de valoarea logică a condiţiei testate cu ajutorul instrucţiunii IF poate conţine la rîndul ei instrucţiunea IF.

Limitaţi extinderea în adîncime a unei structuri de control alternative la cel mult 3 niveluri de includere, iar în lungime la cel mult o pagină de imprimantă.

Exemple:

```
IF CANTITATE NOT NUMERIC
  THEN
    IF NOT (CANTITATE = 0 OR CANTITATE > 50000)
      THEN
        DISPLAY 'CANTITATE ERONATA'
        MOVE 1 TO V.
```

IF-ul îmbricat permite traducerea structurilor alternative multiple de forma celei din figura 8.1.

Exemplu:

```
IF COD—OPERATIE = 1
    THEN
      PERFORM PROCEDURA—OPERATIA—1
    ELSE IF COD—OPERATIE = 2
           THEN
             PERFORM PROCEDURA—OPERATIA—2
           ELSE
             IF COD—OPERATIE = 3
```

```
             THEN
               PERFORM PROCEDURA-OPERATIA-3
             ELSE
               PERFORM PROCEDURA-EROARE.
*         ENDIF
```

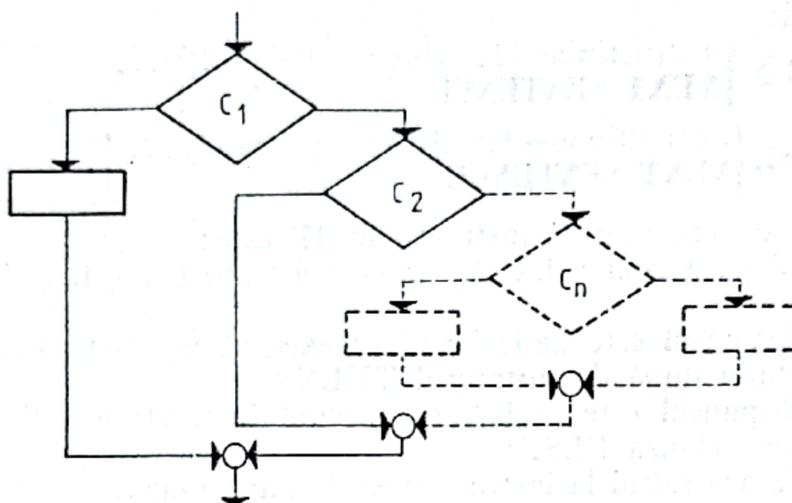


Fig. 8.1.

Observaţie: Pentru a facilita înţelegerea programului se sugerează gruparea instrucţiunilor de prelucrare în secvenţe (proceduri) distincte apelate prin instrucţiunea PERFORM

De asemenea la închiderea instrucţiunii IF se mai poate folosi linia comentariu ENDIF (care conţine caracterul * în coloana 7).

8.18 *Î.* Traduceţi în instrucţiuni COBOL schema logică din figura 8.2. folosind pentru efectuarea calculelor instrucţiunea COMPUTE . . .
. . . .
. . . .

R.

```
ÎNCEPUT.
    IF A = B
        THEN
          COMPUTE A = E * F
        ELSE
          IF I >
             THEN
               COMPUTE C = A / G
             ELSE
               COMPUTE H = M - N.
*
*   ENDIF
```

8.19 *Clauza REDEFINES*, permite ca aceeaşi zonă de memorie să fie alocată la două cîmpuri (elementare sau grupate) diferite.

Exemple:

```
a)  02    CANTITATE PIC 9(5)V99.
    02    R-CANTITATE REDEFINES CANTITATE PIC X(7).
```

b) 02 DATA—SISTEM PIC X(6).
02 DATA—SISTEM—R REDEFINES DATA—SISTEM.
03 ZI PIC 99.
03 LUNA PIC 99.
03 AN PIC 99.

Formatul general al clauzei REDEFINES este:
număr—de—nivel nume—dată—1 **REDEFINES** nume—dată—2

unde:

• *nume—dată—1/nume—dată—2* sînt două cîmpuri de aceeași lungime cărora li se atribuie aceeași zonă de memorie; structura și natura lor poate să difere.

În exemplul *a* cîmpurile R—CANTITATE și CANTITATE au aceeași lungime (7 locații) dar natură diferită.

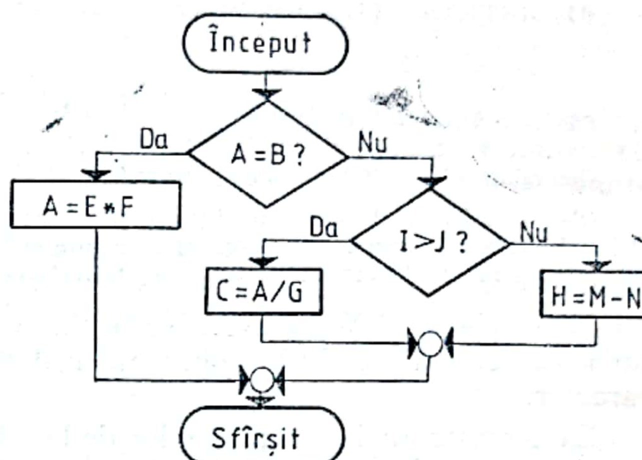


Fig. 8.2.

Rubrica de descriere a datei *nume—dată—2* succede celei de descriere a datei *nume—dată—1* și apare la același număr de nivel. Între rubricile de descriere a celor două date nu pot apare decît rubrici care descriu date subordonate.

8.20 *Ω.* Fie machetele (fig. 8.3).
Se cere utilizarea clauzei REDEFINES pentru a descrie elementele celor două machete ca formînd un singur articol (de la o machetă la alta diferă semnificația acordată caracterelor de la 2—30).

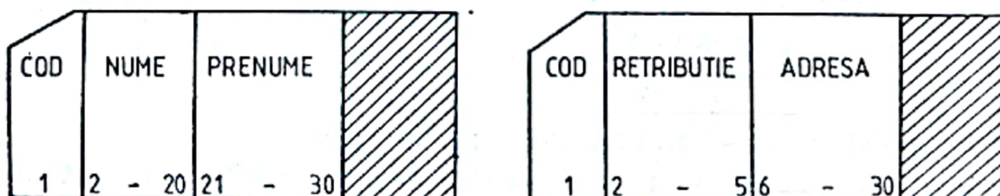


Fig. 8.3.

R. 01 ARTICOL.
02 COD PIC 9.
02 MACHETA—1.
03 NUME PIC X(19).
03 PRENUME PIC X(10).
02 MACHETA—2 REDEFINES MACHETA—1.
03 RETRIB—LUNARĂ PIC 9(4).
03 ADRESA PIC X(25).

8.21 În general, în cazul în care un fişier conţine articole cu structuri diferite (cu aceeaşi lungime sau cu lungime diferită), fiecare tip de articol se descrie la o rubrică de nivel 01.

Exemplu:

```
FD FIŞIER—BANDA
   RECORDING V
   LABEL RECORD STANDARD.
01 ARTICOL—TIP—1.
   .
   .
   .
01 ARTICOL—TIP—2.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Compilatorul COBOL rezervă pentru fişierul FIŞIER—BANDA o singură zonă—articol.

Observaţie În cazul în care un fişier conţine mai multe tipuri de articole folosirea clauzei REDEFINES la rubricile de nivel 01 este interzisă întrucît redefinirea este implicită; dacă rubricile de nivel 01 apar în secţiunea WORKING—STORAGE ele pot fi redefinite.

8.22 *Instrucţiunea EXAMINE*, se utilizează pentru numărarea apariţiilor unui caracter în cadrul valorii unei date şi/sau înlocuirea lui cu un alt caracter.

Ea permite analiza caracterelor de la stînga la dreapta şi după caz contorizarea numărului de apariţii al unui caracter sau înlocuirea lui cu un alt caracter.

Exemple

a) *Starea iniţială*: *ZONA—1*

|ь|ь|ь|ь|5|3|4|

EXAMINE ZONA—1 REPLACING LEADING ' ' BY '0' sau
EXAMINE ZONA—1 REPLACING LEADING SPACES BY ZEROS

Stare finală: |0|0|0|0|5|3|4|

b) *Stare iniţială*: ZONA—1

| | | |1|2| |5|

EXAMINE ZONA—1 REPLACING ALL 'ь' BY '0'

Stare finală: ZONA—1

|0|0|0|1|2|0|5|

Instrucţiunea EXAMINE are două formate:

- *formatul—1*, pentru realizarea numărării şi/sau înlocuirii

EXAMINE nume—dată—1

TALLING { **ALL** / **LEADING** / **UNTIL FIRST** } { nume—dată—2 / literal—1 }

[**REPLACING BY** { nume—dată—3 / literal—2 }]

• *formatul—2*, pentru realizarea înlocuirii unui anumit caracter:

EXAMINE nume—dată—1

REPLACING {**ALL** / **LEADING** / **[UNTIL] FIRST**} {nume—dată—2 / literal—1}

BY {nume—dată—3 / literal—2}

unde:

• *nume—dată—1* desemnează cîmpul ale cărui valori sînt analizate (şi are natură numerică sau nenumerică şi reprezentare un caracter pe un octet);

• *literal—1*, *literal—2*, sînt literale (numerice, nenumerice sau constante figurative) formate dintr-un singur caracter;

• *nume—dată—1*, *nume—dată—2*, desemnează cîmpuri ale căror valori sînt formate dintr-un singur caracter şi au aceiaşi categorie cu literalele;

• *opţiunea ALL* indică numărarea peste tot a caracterelor desemnate prin *literal—2/nume—dată—3*, în cazul formatului—1, şi/sau înlocuirea peste tot cu caracterul desemnat prin *literal—2/nume—dată—3* în cazul formatului —2;

• *opţiunea LEADING* limitează domeniul de acţiune al instrucţiunii EXAMINE la începutul datei *nume—dată—1*;

• *opţiunea UNTIL FIRST*, limitează domeniul de acţiune al instrucţiunii EXAMINE la prima apariţie a caracterului căutat;

• *opţiunea FIRST*, precizează înlocuirea primei apariţii a caracterului căutat.

La datele numerice de calcul nu se indică folosirea opţiunii ALL deoarece aceasta poate duce la obţinerea unor rezultate eronate. Căutarea şi/sau numărarea unui anumit caracter are loc de la stînga la dreapta.

8.23 *Î.* Formulaţi instrucţiunea de examinare a valorilor cîmpului CANTITATE în scopul înlocuirii spaţiilor nesemnificative (din stînga) cu zerouri. Dar dacă am vrea ca înlocuirea să se realizeze peste tot . . . .

R. EXAMINE CANTITATE REPLACING LEADING SPACES BY ZERO;
EXAMINE CANTITATE REPLACING ALL ' ' BY '0'

8.24 *Î.* Literal—1/nume—dată—2 şi respectiv literal—2/nume—dată—3 pot desemna . . . .

R. Un singur caracter

8.25 Rezultatul obţinut prin numărarea apariţiilor unui caracter căutat este memorat în format binar într-un registru special identificat prin cuvîntul rezervat TALLY. De obicei pentru a fi editat, conţinutul registrului TALLY trebuie atribuit unui cîmp reprezentat în cod EBCDIC.

Exemplu:

```
02  ZONA-EBCDIC  PIC 99.
02  ZONA  PIC X(9)  VALUE '=========='.
.
.
.
     EXAMINE ZONA TALLYING ALL '='
     MOVE TALLY TO ZONA-EBCDIC
     DISPLAY 'NUMĂR APARIȚII = ' ZONA-EBCDIC
```

8.26 ∂. Completați în tabelul 8.2 efectul produs de executarea instrucțiunii EXAMINE.

	Instrucțiunea EXAMINE	Valoarea datei examinate		Valoarea registrului TALLY
		Înainte	După	
1	EXAMINE ZONA TALLYING UNTIL FIRST 'B'	BARBU		
2	EXAMINE ZONA TALLYING ALL 'O'	COBOL		
3	EXAMINE ZONA TALLYING ALL 'O' REPLACING BY 'b'	COBOL		
4	EXAMINE ZONA REPLACING LEADING 'C' BY 'Z'	CCCAD		
5	EXAMINE ZONA REPLACING FIRST 'O' BY 'A'	ORAD		

R
1) BARBU, 0
2) COBOL, 2
3) C B L, 2
4) ZZZAD, -
5) ARAD, -

8.27 Instrucțiunea EXAMINE este deseori utilizată în practică pentru înlocuirea spațiilor nesemnificative cu zero.

Se știe că la perforarea datelor numerice în cartele, cînd numărul cifrelor semnificative este mai mic decît lungimea cîmpului repartizat prin machetă, operatorul optează pentru a lăsa spații libere (după ce a realizat alinierea în conformitate cu marca zecimală).

Pentru a aduce valoarea datei la valoarea cerută unui operand numeric care participă în calcule se impune înlocuirea spațiilor nesemnificative. În acest scop data numerică de calcul se redefinește pentru a-i da natură alfanumerică iar asupra ei se aplică instrucțiunea EXAMINE. Valoarea rezultată în urma exeminării poate fi supusă operației de testare pentru a vedea dacă ea este de natură numerică.

Exemplu:

```
02  CANTITATE PIC 9(5)V99.
02  R-CANTITATE REDEFINES CANTITATE PIC X(7).
.
.
.
     EXAMINE R-CANTITATE REPLACING
                    LEADING SPACES BY ZEROS
     IF CANTITATE NOT NUMERIC
                    THEN
                              MOVE 1 TO V
                              DISPLAY 'CANTITATE NENUMERICĂ' .
```

8.28 *Î*. Dați variante de realizare a testului de clasă pentru cîmpul STOC—NORMAT de natură numerică și lungime 6 caractere.

. . . .
. . . .
. . . .

R.

a)

```
02  STOC—NORMAT  PIC 9(6).
.
.
.
IF STOC—NORMAT NOT NUMERIC
           THEN
              DISPLAY 'STOC NORMAT NENUMERIC'.
```

b)

```
02  STOC—NORMAT  PIC 9(6).
02  STOC—NORMAT—R REDEFINES STOC—NORMAT
          PIC X(6).
.
.
.
EXAMINE  STOC—NORMAT—R  REPLACING  LEADING
                        SPACES BY ZEROS
IF STOC—NORMAT NOT NUMERIC
         THEN
             DISPLAY 'STOC NORMAT NENUMERIC'.
```

Pentru verificarea cunoștințelor, testul 2, pagina 350

LECȚIA a 9-a DIVIZIUNEA ENVIRONMENT: COMPLETARE

- paragraful SPECIAL—NAMES
- fraza SELECT; caz fișier memorat pe bandă magnetică (clauza RESERVE)
- fraza SELECT; caz fișiere memorate pe discuri magnetice
  - clauza ORGANIZATION
  - clauza RECORD KEY
  - clauza SYMBOLIC KEY
  - clauza ACTUAL KEY
- paragraful I—O CONTROL
  - modul de tratare MOVE
  - modul de tratare LOCATE
  - clauza APPLY
  - clauza SAME
  - clauza MULTIPLE FILE

9.1 Această lecție este destinată să vă inițieze în descrierea caracteristicilor și a modului de tratare a fișierelor în general și a fișierelor pe suporturi magnetice în special.

9.2 *Î.* Un fișier pe bandă magnetică poate fi organizat numai a) . . . . și exploatat în acces b) . . . . Un fișier pe disc magnetic poate fi organizat c) . . . . și exploatat în acces d) . . . .

R. a) secvențial,
b) secvențial
c) secvențial, secvențial—indexat, selectiv
d) secvențial, direct

9.3 *Î.* În cazul fișierelor cu organizare secvențială—indexată articolele care îl compun sînt . . . . după valorile unui cîmp numit . . . .

R. — ordonate
— cheie de identificare

9.4 *Î.* Suportul folosit pentru memorarea fișierelor cu organizare secvențială—indexată trebuie să fie . . . . și se împarte în trei zone numite . . . .

R. — adresabil
— zona principală, zona de depășire și zona tabelei de indecși

9.5 *Î.* Tabela de indecși servește pentru . . . .
. . . .

R. a realiza corespondența între valorile cîmpului cheie și *unitățile de adresare* la care articolele sînt memorate (pagină, cilindru, volum)

9.6 *Î.* Principalele operații cu fișierele care au organizare secvențială–indexată sînt: . . . .

R. — *creare* și se poate realiza numai în acces secvențial utilizînd în intrare un fișier cu articole ordonate în funcție de valorile crescătoare ale cheii de identificare;
— *consultare* și se poate realiza în acces secvențial sau direct;
— *actualizare* și se realizează, în general, în acces direct.

9.7 *Î.* La consultarea în acces direct a unui fișier cu organizare secvențială–indexată programatorul trebuie să precizeze . . . . care face parte din structura articolului și . . . .

R. — cheia de identificare
— cheia de adresare

9.8 *Î.* Fișierele cu organizare selectivă se definesc ca fiind . . . . grupate în . . . . . . pe baza unei funcții de . . . .
Articolele dintr-o clasă sînt . . . .

R. — o colecție de articole
— clase
— randomizare
— sinonime

9.9 *Î.* Identificarea articolelor dintr-o clasă se realizează pe baza . . . .
. . . .

R. unei chei de identificare

9.10 Algoritmul de randomizare trebuie astfel ales încît să asigure o repartizare cît mai uniformă a articolelor în căsuțe (numim căsuță spațiul de memorare a articolelor dintr-o clasă).

9.11 *Î.* Principalele operații cu fișierele care au organizare selectivă sînt:
. . . .
toate se realizează în acces . . . .

R. — creare, consultare, actualizare
— direct

Răspunsurile sînt corecte

DA → 9.12.
NU ↶ Lecția 2

9.12 *Î.* Din lecția 4 ați reținut că diviziunea ENVIRONMENT cuprinde două secțiuni și anume:
. . . .
. . . .

R. CONFIGURATION SECTION.
INPUT–OUTPUT SECTION.

9.13 Secțiunea CONFIGURATION cuprinde mai multe paragrafe (toate avînd caracter opțional):

SOURCE—COMPUTER. paragraf—comentariu.

OBJECT—COMPUTER. paragraf—comentariu.

SPECIAL—NAMES. clauze.

9.14 *Paragraful SPECIAL—NAMES* se utilizează în programele în care se editează situații la imprimantă. Prin clauzele sale programatorul specifică unele convenții de editare cum ar fi:

— imprimarea unui anumit semn monetar, CURRENCY SIGN IS literal

Remarcă: Simbolul monetar standard este $. Dacă dorim să-l schimbăm cu o altă literă exemplu L (leu), atunci în cadrul acestui paragraf scriem:

CURRENCY SIGN IS 'L'

— imprimarea virgulei în locul punctului zecimal folosind clauza:

DECIMAL—POINT IS COMMA.

La editare, în principiu, numerele întregi sînt separate de cele zecimale prin punctul zecimal (exemplu 1578.99). Dacă dorim ca ele să fie separate prin virgulă zecimală (1578,99) vom utiliza în program clauza DECIMAL—POINT IS COMMA.

Observație: Paragraful SPECIAL—NAMES poate cuprinde și alte clauze, care fiind mai puțin utilizate nu au fost descrise.

9.15 **Exemplu** complet:

Se dorește ca într-o situație la imprimantă să se înlocuiască punctul cu virgula zecimală iar semnul monetar $ cu L.

Secțiunea CONFIGURATION poate avea structura:

```
CONFIGURATION SECTION.
SOURCE—COMPUTER. FELIX C—256.
OBJECT—COMPUTER. FELIX C—256.
SPECIAL—NAMES. CURRENCY SIGN IS 'L'
               DECIMAL—POINT IS COMMA.
```

9.16 *Î.* Secțiunea CONFIGURATION furnizează informații în legătură cu:

. . . .

R. — calculatorul folosit în operația de compilare a programului sursă;
— calculatorul folosit în operația de execuție a programului obiect;
— anumite convenții folosite la imprimare.

9.17 Secțiunea INPUT—OUTPUT cuprinde două paragrafe și anume:

— FILE—CONTROL, în care se descriu caracteristicile privind organizarea și accesul;

— I—O—CONTROL, în care se precizează modul de tratare a fișierelor.

9.18 Î. Pentru a descrie caracteristicile unui fişier se va utiliza fraza . . .
. . . .

R. SELECT

9.19 Cînd fişierul este memorat pe bandă magnetică structura simplificată a frazei SELECT este:

SELECT nume—fişier **ASSIGN** TO {idex—1 | idex—2}

[**RESERVE** {**NO** | întreg} ALTERNATE {AREA | AREAS}].

unde:

- *idex—1*, poate fi o literă de la A la Z;
- *idex—2*, poate fi o literă de la A la Z urmată de codul MT pentru indicarea unităţii de bandă magnetică.

Exemple: (fig. 9.1):

SELECT	F—BAND—1	ASSIGN	A.
sau			
SELECT	F—BAND—1	ASSIGN	AMT.
SELECT	F—BAND—2	ASSIGN	B.
sau			
SELECT	F—BAND—2	ASSIGN	BMT.

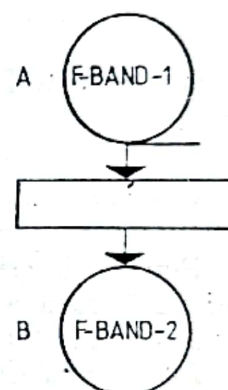


Fig. 9.1.

9.20 *Clauza RESERVE*, serveşte pentru a indica suplimentarea sau suprimarea numărului zonelor tampon utilizate în tratarea fişierelor. Pentru a înţelege această clauză să ne reamintim ce înseamnă zonă—tampon (BUFFER): fig. 9.2. Timpul afectat operaţiei de intrare/ieşire este mare, fapt pentru care unitatea centrală trebuie să aştepte, după efectuarea operaţiilor de prelucrare, pînă cînd un nou bloc este adus din memoria auxiliară în zona tampon iar articolul depus în zona articol. Pentru acest motiv se indică folosirea mai multor zone tampon.

În timp ce se realizează o operaţie de intrare/ieşire care implică prima zonă tampon, datele din celelalte zone pot fi prelucrate (fig. 9.3).

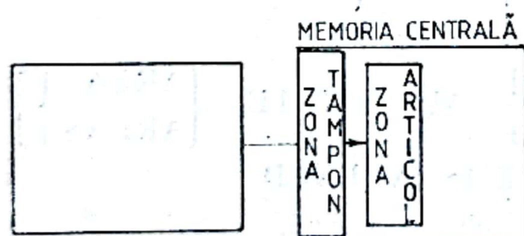


Fig. 9.2.

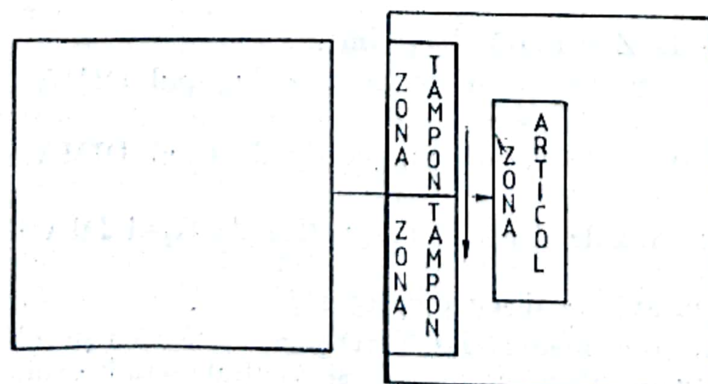


Fig. 9.3.

Observaţie: Compilatorul afectează pentru fiecare fişier cu organizare secvenţială două zone tampon.

Cînd în clauza RESERVE se foloseşte opţiunea *NO* înseamnă că din cele două zone tampon afectate în mod obişnuit se suprimă una. Opţiunea „întreg" precizează numărul de zone tampon de suplimentare (cînd valoarea întregului este mai mare decît zero).

Exemplu:

	Nr. zone tampon afectate
RESERVE NO	1
RESERVE 0	1
RESERVE 4	6

9.21 *Î*. Scrieţi fraza SELECT pentru un fişier pe bandă magnetică numit F—BANDA—MAGNETICA de idex C.
Pentru optimizarea prelucrării se cere afectarea unui număr de 8 zone tampon

. . . .

. . . .

R. SELECT F—BANDA—MAGNETICA
ASSIGN CMT
RESERVE 6 ALTERNATE AREAS.

Observaţie: „Întreg" nu poate lua decît valori pe intervalul {0, 14}.

9.22 Pentru descrierea caracteristicilor unui fişier cu organizare secvenţială—indexată fraza SELECT are structură diferită, în funcţie de operaţia efectuată asupra fişierului şi de modul de realizare a acesteia.

9.23 *Fraza SELECT* pentru crearea unui fişier cu organizare secvenţială — indexată are formatul:

SELECT nume—fişier **ASSIGN** TO {idex—1 | idex—2}
[**RESERVE** {**NO** | întreg} ALTERNATE {AREA | AREAS}]
ORGANIZATION MODE IS **INDEXED**
RECORD KEY nume—cheie—articol.

unde:

● *Idex—2,* o literă de la A la Z urmată de codurile:

MD, pentru a desemna o unitate de discuri magnetice de tipul DIMAS (MD 25);

AD, pentru a desemna o unitate de discuri magnetice de tipul DIMAS (MD 50);

RD, pentru a desemna o unitate de discuri magnetice de tipul DIAM (disc amovibil);

DK, pentru a desemna o unitate de discuri oarecare.

9.24 *Clauza ORGANIZATION* prin opţiunea INDEXED precizează că fişierul selectat pentru operaţia de creare va avea organizare secvenţială—indexată.

9.25 Deoarece crearea nu se poate realiza decît printr-o dispunere secvențială a articolelor (etapă în care se constituie și tabelele cu indecși), clauza ACCESS care indică modul de acces, este opțională. În lipsa acestei clauze compilatorul consideră accesul secvențial.

9.26 *Clauza RECORD KEY* specifică numele cîmpului (elementar sau grup) care face parte din articol și reprezintă cheia de identificare.

Exemplu:
```
RECORD KEY COD—MATERIAL
```

9.27 **Exemplu** complet de frază SELECT pentru fișier cu organizare secvențială—indexată utilizat în acces secvențial, cîmpul COD—MATERIAL constituie cheia de identificare:
```
8       12
        SELECT FISIER—DISC—SI ASSIGN AMD
                ORGANIZATION INDEXED
                RECORD KEY COD—MATERIAL.
```
iar în cadrul diviziunii DATA:
```
DATA DIVISION.
FILE SECTION.
FD FIȘIER—DISC—SI . . . . . . . . . . . . . . .
01  ARTICOL.
  02  COD—MATERIAL      PIC 9(7)  COMP—3.
```

9.28 *Fraza SELECT* pentru consultarea unui fișier cu organizare secvențială—indexată are formatul:

SELECT nume—fișier **ASSIGN** TO { idex—1 | idex—2 }

[**REZERVE** { **NO** | întreg } ALTERNATE { AREA | AREAS }]

ORGANIZATION MODE IS **INDEXED**

[**ACCESS** MODE IS { **SEQUENTIAL** | **RANDOM** }]

RECORD KEY nume—data—1

SYMBOLIC KEY nume—dată—2.

- *Clauza ACCESS* (opțională în cazul consultării secvențiale) precizează modul de acces la articolele fișierului; opțiunea RANDOM indică accesul direct.
- *Clauza SYMBOLIC* precizează prin *nume—dată—2* cîmpul (elementar sau grup) care constituie *cheia de acces* la articolele fișierului (el poate fi descris în secțiunea WORKING—STORAGE sau în secțiunea FILE; în cel din urmă caz face parte din structura altui articol decît cel corespunzător fișierului cu organizare secvențială—indexată).

Exemplu de frază SELECT pentru consultare prin acces direct:
```
SELECT  FISIER—SECVENTIAL—INDEXAT
        ORGANIZATION INDEXED
        ACCESS RANDOM
        RECORD KEY COD—MATERIAL
        SYMBOLIC KEY CHEIE—SIMBOLICA.
```

iar în cadrul diviziunii DATA:

```
DATA DIVISION.
FILE SECTION.
FD  FIŞIER—SECVENŢIAL—INDEXAT ......
01    ARTICOL.
  02    COD—MATERIAL  9(7)  COMP—3.
. . . . . . . . . . . . . . . . . . . . . . . . . .
WORKING—STORAGE SECTION.
77    CHEIE—SIMBOLICA  9(7) COMP—3.
.
.
.
```

9.29 Fraza SELECT pentru operații de actualizare a fișierelor cu organizare secvențială—indexată are structura asemănătoare celei pentru consultare.

9.30 *Î.* Scrieți fraza SELECT pentru consultare în acces direct a fișierului FIȘIER—PRODUSE care are organizare secvențială—indexată și indexul BAD. Cheia de identificare a articolului este COD—PRODUS iar cheia de acces CHEIE—ACCES.

. . . .

. . . .

. . . .

. . . .

R.
```
SELECT FIŞIER—PRODUSE ASSIGN BAD
                      ORGANIZATION INDEXED
                      ACCESS RANDOM
                      RECORD KEY COD—PRODUS
                      SYMBOLIC KEY
                           CHEIE—ACCES.
```

9.31 În varianta în care *cheia de adresare* este descrisă în secțiunea WORKING—STORAGE ordinea de realizare a operațiilor în vederea consultării directe este următoarea (fig 9.4):

— se citește un articol din fișierul pilot (fig. 9.5) care determină ce articol trebuie citit din fișierul cu organizare secvențială—indexată;

— se atribuie valoarea, care desemnează aricolul ce trebuie citit din fișierul cu organizare secvențială—indexată, cîmpului cheie de acces;

— se realizează operația de citire a articolului a cărui *cheie de identificare* este egală cu *cheia de acces* (se subînțelege că este vorba de valorile celor două cîmpuri).

9.32 Pentru descrierea caracteristicilor unui fișier cu organizare selectivă se folosește fraza SELECT cu următorul format:

```
SELECT nume—fişier ASSIGN TO {idex—1}
                             {idex—2}

        ORGANIZATION DIRECT
        ACCESS RANDOM
        SYMBOLIC KEY nume—dată—1
        ACTUAL KEY   nume—dată—2.
```

Clauza ACTUAL desemnează un cîmp căruia i se atribuie adresa căsuței în care este memorată clasa din care face parte articolul în cauză (vezi lecția 2).

9.33 2. Completați tabelul 9.1 cu elementele corespunzătoare.

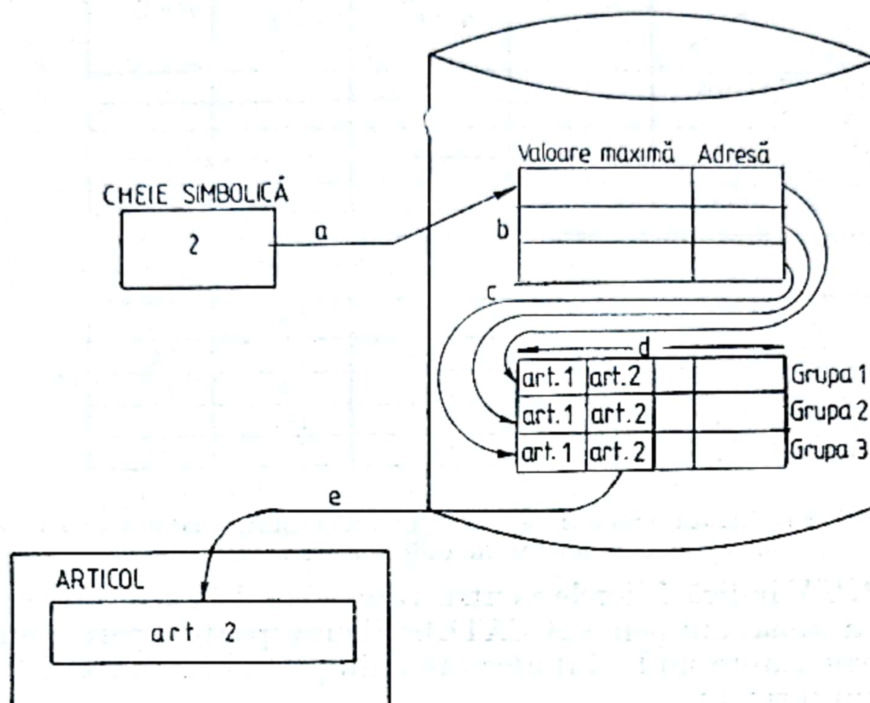


Fig. 9.4.

9.34 *Paragraful I—O—CONTROL* se folosește pentru a furniza informații în legătură cu modul de tratare a fișierelor. Toate clauzele din componența acestui

9.35 paragraf sînt opționale. Fișierele pot fi tratate în două moduri:

a) — *Modul MOVE*, cînd compilatorul alocă fiecărui fișier cîte o zonă articol (fig. 9.6).

S.G.F.-ul realizează, în mod automat, trecerea articolului în cauză din zona tampon în zona articol.

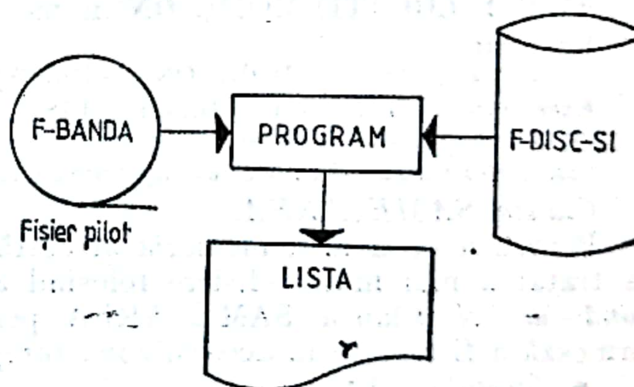
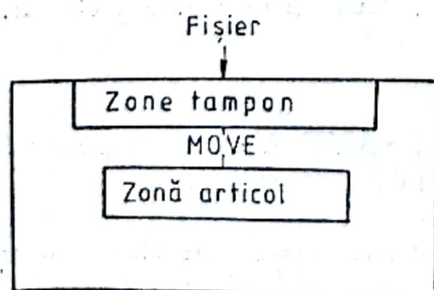


Fig. 9.5.

b) — *Modul LOCATE*, cînd compilatorul nu alocă zone articol, prelucrarea articolelor făcîndu-se direct în zona tampon (fig. 9.7).



←Fig. 9.6.

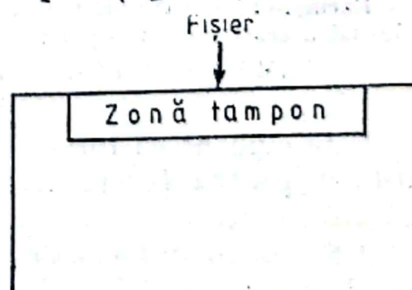


Fig. 9.7.→

Clauza \ Moduri de organizare și acces	Secvențială	Secvențial-indexată (cu acces secvențial)	Secvențial-indexată (cu acces direct)	Selectivă (cu acces direct)
ORGANIZATION				
ACCESS				
RECORD KEY				
SYMBOLIC KEY				
ACTUAL KEY				

R.

ORGANIZATION	opțională	DA	DA	DA
ACCESS	opțională	opțională	DA	DA
RECORD KEY	NU	DA	DA	NU
SYMBOLIC KEY	NU	NU	DA	DA
ACTUAL KEY	NU	NU	NU	DA

Observație: Cînd se realizează o prelucrare în mod LOCATE programatorul trebuie să urmărească cu atenție asigurarea alinierii datelor în cadrul articolului *.

9.36 *Clauza APPLY* indică fișierele pentru care accesul la articole se realizează direct în zona tampon (LOCATE). Clauza poate apare numai la fișiere secvențiale cărora nu le sînt afectate unitățile standard ale sistemului și are următorul format:

APPLY LOCATE MODE **ON** nume—fișier—1 [nume—fișier—2]...

Exemplu:

APPLY LOCATE MODE ON F—BANDA

Exemplu: fie FIȘIER—BANDA—1 și FIȘIER—BANDA—2 două fișiere supuse prelucrării folosind aceiași zonă tampon:

Observație: Clauza APPLY este obligatorie cînd se utilizează fișiere cu organizare nedefinită.

Clauza SAME AREA

Pentru a economisi memoria afectată unui program se poate proceda la tratarea mai multor fișiere folosind aceeași *zonă—tampon* sau aceeași *zonă—articol.* Clauza SAME AREA permite să se indice fișierele care urmează a fi tratate în aceeași zonă tampon și/sau în aceeași zonă articol.

• *formatul—1* :

SAME AREA FOR nume—fișier—1 {nume—fișier—2}...

• *formatul—2* :

SAME RECORD AREA FOR nume—fișier—1 {nume—fișier—2}...

Exemplu: fie FISIER-BANDA-1 și FISIER-BANDA-2 două fișiere supuse prelucrării folosind aceeași zonă tampon:

SAME AREA FOR FIȘIER—BANDA—1
FIȘIER—BANDA—2.

Este evident că formatul al doilea al clauzei SAME AREA nu poate fi utilizat pentru fișiere tratate în mod LOCATE.

* Sînt cazuri cînd adresele de memorie, afectate datelor cu reprezentare binară sau în virgulă mobilă, nu sînt multiplu de 2, 4 sau 8.

9.37 *Indicații practice*: se recomandă utilizarea clauzei SAME, cînd în același program se face crearea, actualizarea și, după caz, consultarea unui fișier, deci cînd se utilizează trei fișiere distincte, cu idexuri și nume diferite, dar cu aceeași zonă tampon.

Exemplu:

```
SELECT FISIER—1        ASSIGN A.
SELECT FISIER—2        ASSIGN B.
. . . . . . . . . . . . . . . . . . . . .
SAME AREA FOR          FISIER—1
                       FISIER—2
```

9.38 În condițiile folosirii fișierelor pe bandă magnetică se poate organiza memorarea datelor după schema multifișierelor. Clauza MULTIPLE FILE permite declararea unei structuri multifișier (fișierele sînt înregistrate pe suport în ordinea în care apar).

Exemplu de multifișier pe bandă magnetică (fig. 9.8):

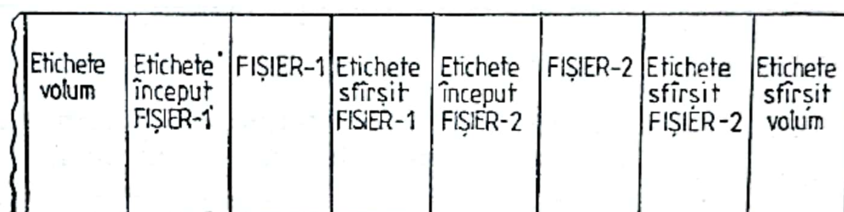


Fig. 9.8.

Pot fi exploatate toate fișierele sau numai o parte din ele (tratarea se realizează însă în mod secvențial).

```
FILE—CONTROL.
    SELECT  FISIER—BANDA—A          ASSIGN A.
    SELECT  FISIER—BANDA—B          ASSIGN B.
    SELECT  FISIER—BANDA—C          ASSIGN C.
I—O—CONTROL.
    SAME AREA FOR     FISIER—BANDA—A
                      FISIER—BANDA—B
                      FISIER—BANDA—C
    MULTIPLE FILE     FISIER—BANDA—A
                      FISIER—BANDA—B
                      FISIER—BANDA—C.
```

Observație După ce au fost prelucrate datele din FIȘIER—BANDA—A nu trebuie realizată rebobinarea benzii.

9.39 *Î.* Paragraful I—O—CONTROL se utilizează pentru a indica modul de . . . .

R. tratare a fișierelor

9.40 *Î.* Principalele clauze utilizate sînt :

— APPLY care indică . . . .
. . . .

— SAME AREA care indică . . . .

. . . .
— MULTIPLE FILE . . . .
care indică . . . .

R. Comparați răspunsurile cu cele din paragrafele 9.34—9.38

9.41 *Î.* Un fișier poate fi tratat:
— în mod MOVE cînd . . . .
. . . .
— în mod LOCATE cînd . . . .
. . . .

R. Comparați răspunsurile cu cele din paragrafele 9.33—9.38
Ele sînt corecte?

NU → Lecția 10
DA ↰ Lecția 9.1

LECȚIA a 10-a RUBRICI DE DESCRIERE A FIȘIERELOR ȘI DATELOR: FORMATE DEZVOLTATE

- structura rubricii FD
  - clauza BLOCK CONTAINS
  - clauza RECORD CONTAINS
  - clauza LABEL
- rubrici de descriere a datelor
  - clauza PICTURE
  - șabloane
  - clauza VALUE

10.1 *Î.* Așa cum s-a prezentat în lecția 5 diviziunea de date cuprinde 4 secțiuni (opționale în program) și anume:

. . . .

R. FILE SECTION
WORKING—STORAGE SECTION
LINKAGE SECTION
REPORT SECTION

10.2 *Î.* În cadrul secțiunii FILE se descriu a) . . . . iar în secțiunea WORKING—STORAGE b) . . . .

R. a) fișierele și articolele asociate acestora
b) datele intermediare, care pot fi independente (apar la numărul de nivel 77) sau grupate (cele independente trebuie să le preceadă pe cele grupate)

10.3 Descrierea fișierelor pe suporturi magnetice se realizează cu următoarea structură a rubricii FD:

FD nume—fișier

[**RECORDING** MODE IS { **F** | **V** | **U** }]

LABEL { **RECORD** IS | **RECORDS** ARE } { **OMITTED** | **STANDARD** | etichetă—utilizator }

[**BLOCK** CONTAINS întreg—1 [**TO** întreg—2] {CHARACTERS / **RECORDS**}]

[**RECORD** CONTAINS întreg—3 [**TO** întreg—4] CHARACTERS]

[**DATA** {**RECORD** IS / **RECORDS** ARE} nume—articol].

10.4 În condițiile folosirii suporturilor magnetice o înregistrare fizică poate conține una sau mai multe articole (înregistrări logice) fig. 10.1.

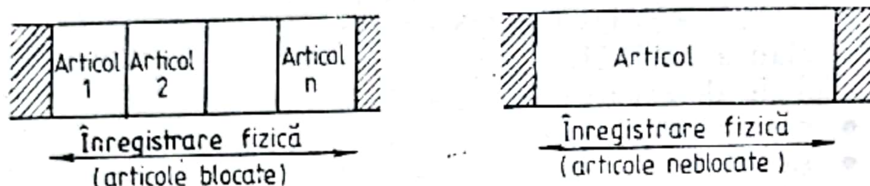


Fig. 10.1.

Clauza BLOCK CONTAINS utilizată în program precizează:
— numărul articolelor dintr-un bloc, cînd se folosește opțiunea RECORDS

Exemplu:
BLOCK CONTAINS 50 RECORDS, unde 50 reprezintă numărul de articole memorate într-o înregistrare fizică;

— dimensiunea zonei tampon, cînd se folosește opțiunea CHARACTERS (opțiune implicită).

Exemplu:
BLOCK CONTAINS 1024 CHARACTERS.

10.5 Informațiile furnizate de către clauza BLOCK CONTAINS servesc compilatorului pentru a determina dimensiunea zonei tampon; lungimea blocului (a zonei tampon) se calculează astfel:

— *pentru fișiere cu organizare secvențială*:

- cînd articolele au lungime fixă $RSF = 8 + n^{*} (RCS^{**} + 1)$
- cînd articolele au lungime variabilă

$$RFS = 8 + n(RCS + 5)$$

— *pentru fișierele cu organizare secvențial—indexată*

- cînd articolele au lungime fixă

$$BFS = 8 + n(RCS + 6) + K \begin{cases} n = 1, & K = 2 \\ n > 1, & K = 0 \end{cases}$$

- cînd articolele au lungime variabilă

$$BFS = 8 + n(RCS + 8)$$

— *pentru fișierele cu organizare selectivă*

* n = factor de blocaj
** RCS = lungimea articolului

- cînd articolele au lungime fixă

$$BFS = 8 + n(RCS + C + 2) + K \quad \begin{cases} n = 1, & K = 8 \\ n > 1, & K = 0 \end{cases}$$

- cînd articolele au lungime variabilă

$$BFS = 8 + n(RCS + C + 4) \quad \begin{cases} n = 1, & K = 6 \\ n > 1, & K = 0 \end{cases}$$

unde: C, lungimea cheii de identificare a articolelor.

10.6 *Opțiunea TO* se utilizează cînd lungimea blocului este variabilă, pentru a indica limita inferioară și respectiv superioară:

BLOCK CONTAINS 15 TO 17 RECORDS

10.7 *Clauza RECORD CONTAINS* este opțională în program, și precizează lungimea zonei—articol:

RECORD CONTAINS 38 CHARACTERS

Cînd în același fișier sînt memorate articole de lungimi diferite, cu ajutorul opțiunii TO se precizează lungimea minimă și respectiv maximă.

10.8 *Clauza LABEL* este obligatorie și se folosește pentru a indica prezența, prin opțiunea STANDARD, sau absența, prin opțiunea OMITTED, a etichetelor care asigură identificarea și protecția fișierelor.

În principiu, fișierelor pe suporturi magnetice li se asociază un sistem de etichete standard.

Suportul bandă magnetică dă posibilitatea programatorului de a adăuga la sistemul de etichete standard etichete de tip utilizator (folosite pentru a crea un sistem și mai riguros de control).

10.9 Fie FIȘIER—BANDA un fișier memorat pe bandă magnetică, avînd înregistrările de lungime fixă și factorul de blocaj 30.

Rubrica FD poate avea structura:

```
FD FISIER—BANDA    RECORDING F
                   BLOCK CONTAINS 30 RECORDS
                   LABEL RECORD STANDARD.
```

10.10 Fie F—DISC un fișier memorat pe disc magnetic avînd înregistrări de lungime variabilă iar factorul de blocaj 50.

Rubrica FD poate avea structura:

```
FD F—DISC          RECORDING V
                   BLOCK CONTAINS 50 RECORDS
                   LABEL RECORD STANDARD.
```

10.11 **Exemplu:** fie fișierul F—MATERIALE memorat pe bandă magnetică căruia i se asociază și etichete utilizator (articolele au lungime variabilă). Rubrica FD și rubricile de date au structura:

```
FD F—MATERIALE RECORDING V
               LABEL RECORD ETICHETA—UT
               BLOCK CONTAINS 40 RECORDS.
```

```
01  ETICHETA—UT.
    02  ANTET                       PIC XXX.
    02  NUMĂR—ETICHETA              PIC 9.
    02  FILLER                      PIC X(76).
01  ARTICOL—BANDA.
    02  COD—MATERIAL                PIC 9(7).
    02  DENUMIRE—MAT                PIC X(4).
    02  UM                          PIC XXX.
    02  CANT                        PIC 9(5)V99.
    02  FILLER                      PIC X(23).
```

10.12 *Î.* Fie F-PRODUSE un fișier pe bandă magnetică despre care se știe că are articole de lungime fixă memorate cîte 80 într-un bloc. Rubrica FD are structura . . . .

R.
```
FD F—PRODUSE  RECONDING F
              LABEL RECORD STANDARD
              BLOCK CONTAINS 80 RECORDS.
```

10.13 Orice rubrică de descriere a fișierelor trebuie să fie urmată de rubricile de descriere a articolului (articolelor).

10.14 *Î.* Formulați rubricile de descriere a articolului cu structura prezentată în fig. 10.2.

COD-PRODUS	DENUMIRE-PRODUS	CANT-STOC
n	a-n	n
7	30	7+2

Fig. 10.2.

R.
```
01    ARTICOL.
  02  COD—PRODUS                    PIC 9(7).
  02  DENUMIRE—PRODUS               PIC X(30).
  02  CANT—STOC                     PIC 9(7)V99.
```

Răspunsul dumneavoastră este corect?

DA → 10.15.
NU ↰ 4.

10.15 Fie ARTICOL—MATERIALE cu macheta din fig. 10.3.

DOCUMENT				GEST	CONTURI				SIMB	UM	CANT	PRET
NUMAR	DATA				DEBIT		CREDIT					
	ZI	LUNA	AN		SINT	ANAL	SINT	ANAL				

Fig. 10.3.

Pentru a facilita descrierea datelor se poate proceda la reprezentarea acestora într-o formă arborescentă (fig. 10.4).

10.16 *Î.* Folosind structura de la paragraful 10.15 descrieți structura articolului indicînd doar datele componente și poziția lor ierarhică. Întrucît DATA este un cuvînt rezervat scrieți în program DATA—DOC.

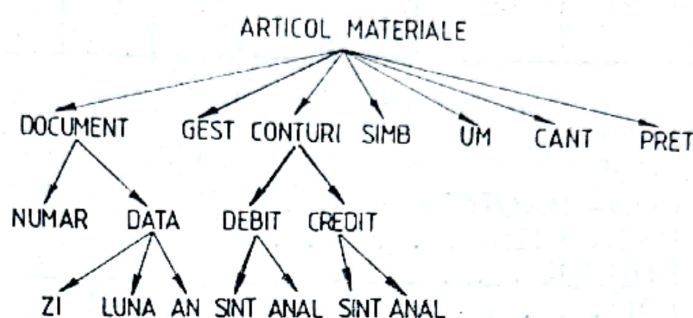


Fig. 10.4

. . . .

R. 01 ARTICOL—MATERIALE.
 02 DOCUMENT.
  03 NUMĂR . . . .
  03 DATA—DOC.
   04 ZI . . . .
   04 LUNA . . . .
   04 AN . . . .
 02 GEST . . . .
 02 CONTURI.
  03 DEBIT.
   04 SINT . . . .
   04 ANAL . . . .
  03 CREDIT.
   04 SINT . . . .
   04 ANAL . . . .
 02 SIMB . . . .
 02 UM . . . .
 02 CANT . . . .
 02 PRET . . . .

10.17 Să luăm acum un alt exemplu în care se precizează că anumite cîmpuri din cadrul articolului nu sînt referențiate (fig. 10.5).

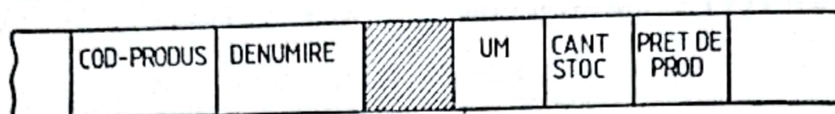


Fig. 10.5.

Descrierea articolului este:

```
01 ARTICOL.
   02 COD—PRODUS . . . . . .
   02 DENUMIRE . . . . . .
   02 FILLER . . . . . .
   02 UM . . . . . .
   02 CANT—STOC . . . . . .
   02 PRET—DE—PROD . . . . . .
```

Structura aleasă de către programator pentru a descrie un articol depinde de utilizarea acestuia în program; pentru același articol, în funcție de program, se pot utiliza diverse structuri.

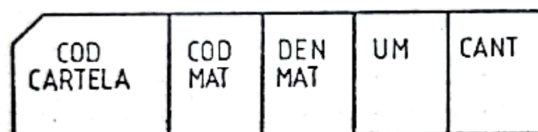


Fig. 10.6.

10.18 Fie structura de articol pentru un fișier pe cartele (fig. 10.6). În condițiile în care programatorul dorește memorarea datelor pe un suport magnetic (mai puțin data privind codul cartelei) poate utiliza la descriere una din variantele:

```
a) 01 CARTELA.
      02 COD—CARTELA . . . . . .
      02 COD—MAT . . . . . .
      02 DEN—MAT . . . . . .
      02 UM . . . . . .
      02 CANT . . . . . .
b) 01 CARTELA.
      02 COD—CARTELA . . . . . .
      02 GRUP—BANDA.
         03 COD—MAT . . . . . .
         03 DEN—MAT . . . . . .
         03 UM . . . . . .
         03 CANT . . . . . .
```

Avantajul celei de a doua variante rezidă în modul de realizare al operațiilor de transfer și respectiv de scriere pe suport.

Descrierea articolului pentru fișierul pe bandă se poate realiza astfel:

```
a) 01 ARTICOL—BANDA.
      02 COD—MAT—B . . . . . .
      02 DEN—MAT—B . . . . . .
      02 UM—B . . . . . .
      02 CANT—B . . . . . .
b) 01 ARTICOL—BANDA PIC X(. . .).
```

iar operaţiile de transfer şi înregistrare pe bandă:

```
   a)  MOVE COD—MAT TO COD—MAT—B
       MOVE DEN—MAT TO DEN—MAT—B
       MOVE UN TO UM—B
       MOVE CANT TO CANT—B
       WRITE ARTICOL—BANDA
   b)  WRITE ARTICOL—BANDA FROM GRUP—BANDA
```

10.19 *Î.* Fişierul PERSONAL memorat pe disc magnetic are articole de lungime fixă şi blocate, factorul de blocaj fiind 25, structura datelor este redată în figura 10.7 (unde LIBER desemnează zona neutilizată).

NUMELE SI PRENUMELE	FUNCTIA	STUDII	ADRESA	LIBER	RETRIBUTIA TARIFARA

Fig. 10.7.

Precizaţi rubricile de descriere a fişierului şi a articolului asociat.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

R.

```
FD PERSONAL   RECORDING F
              BLOCK CONTAINS 25 RECORDS
              LABEL RECORD STANDARD.
01  ARTICOL.
    02  NUME—PRENUME . . . . . .
    02  FUNCŢIA . . . . . .
    02  STUDII . . . . . .
    02  ADRESA . . . . . . .
    02  FILLER . . . . . .
    02  RETRIBUŢIA—TARIFARĂ . . . .
```

10.20 *Numărul de nivel 88* se foloseşte pentru rubrici de date care descriu nume de condiţie. Un nume de condiţie reprezintă un identificator asociat unei valori sau unui ansamblu de valori (interval).

Structura generală a rubricii de descriere a datelor de nivel 88 este:

88 nume—de—condiţie

$$\left\{\begin{array}{l}\textbf{VALUE IS } \text{literal—1}\\ \textbf{VALUE ARE } \text{literal—1}\end{array}\right\}\left[\left\{\begin{array}{c}\textbf{THROUGH}\\ \textbf{THRU}\end{array}\right\}\text{literal—2}\left[\text{literal—3}\left\{\begin{array}{c}\textbf{THROUGH}\\ \textbf{THRU}\end{array}\right\}\text{literal—4}\right]\right]\Big\}$$

Exemple:

```
a)   .
     .
     .
     02    FEL—OPERAŢIE  PIC 9.
        88 INTRĂRI     VALUE 1.
        88 IEŞIRI      VALUE 2.
b)   77    SFÎRŞIT—FIŞIER      PIC XX.
        88 DA—SFÎRŞIT—FIŞIER  VALUE  'DA'.
c)   .
     .
     .
     02    GESTIUNE              PIC 99.
        88 VALOARE—CORECTA  VALUE 1  THRU 15.
```

10.21 Avantajele oferite de utilizarea numelui de condiţie rezultă din maniera de realizare a testului de relaţie.

Spre exemplu, în cazul în care nu se foloseşte numele de condiţie, testul de relaţie pentru a direcţiona procesul de prelucrare conform celor prezentate în figura 10.8 este următorul:

```
IF TIP—OPERATIE = 1
      THEN
         PERFORM PARAGRAF—INTRĂRI
      ELSE IF TIP—OPERATIE = 2
                THEN
                   PERFORM PARAGRAF—IEŞIRI
             ELSE PERFORM PARAGRAF—EROARE.
```

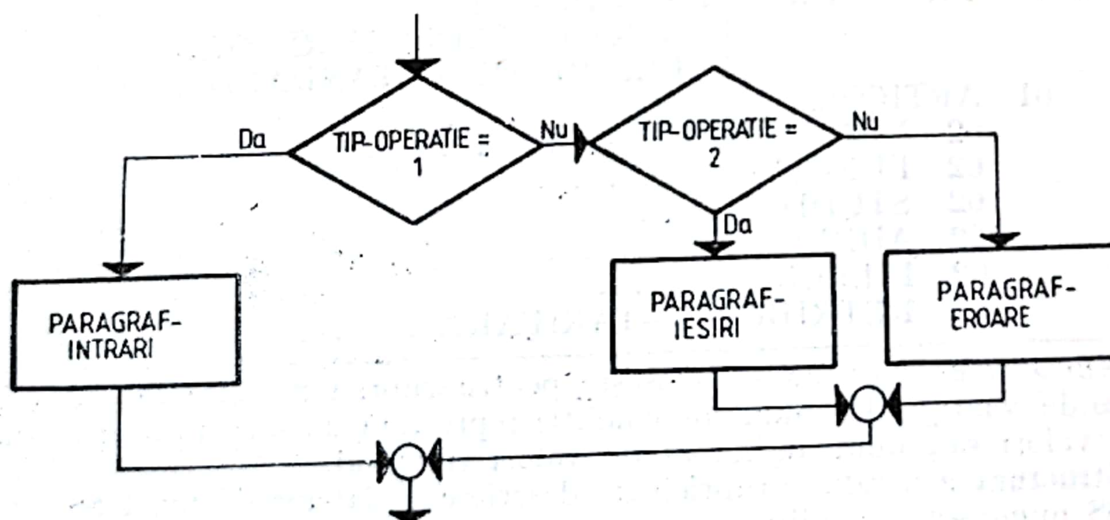


Fig. . .

În condiţiile folosirii rubricilor de nivel 88 (şi prin urmare a numelui—de—condiţie) testul se realizează ca în figura 10.9 iar instrucţiunile COBOL de codificare sînt :

```
IF INTRĂRI
   THEN
        PERFORM PARAGRAF–INTRĂRI
   ELSE IF IEŞIRI
                THEN
                     PERFORM PARAGRAF–IEŞIRI
                ELSE
                     PERFORM PARAGRAF–EROARE.
```

10.22 *Clauza PICTURE*, aşa cum s-a prezentat în lecţiile anterioare precizează cu ajutorul şablonului natura (clasa) şi lungimea datelor.

10.23 *Î.* Clauza PICTURE, apare numai în cadrul rubricilor de descriere a datelor . . . .

Ea permite programatorului să precizeze prin intermediul şablonului . . . .

R. — elementare
— natura şi lungimea datelor

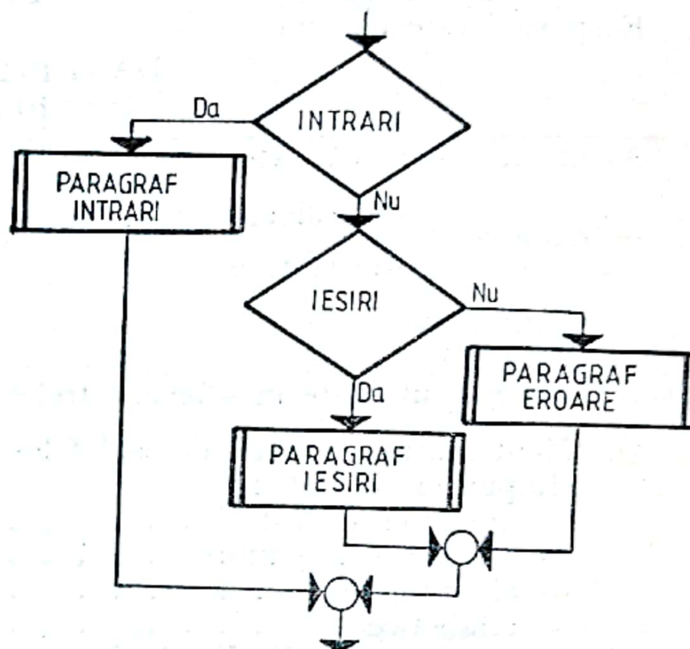


Fig. 10.9.

10.24 La formularea şabloanelor avem în vedere: natura şi lungimea maximă a datelor; iar pentru date numerice de editare caracterele de înserat şi opţiunile de editare.

Fie cîmpul PRET cu următoarea descriere:

02 PRET PIC 9(5)V99.

La analiza acestei rubrici, în memorie, va fi rezervată o zonă de 7 caractere identificate prin cuvîntul utilizator PRET (fig. 10.10).

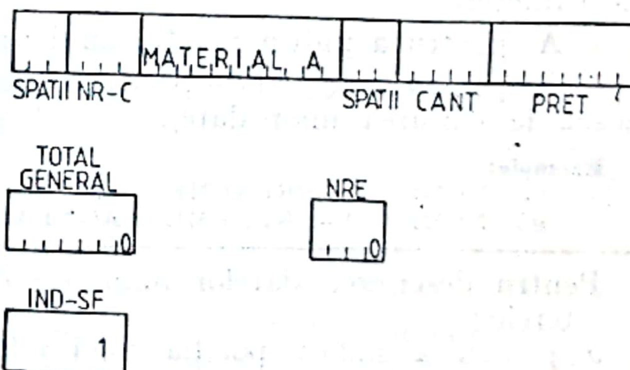


Fig. 10.10.

Î. Scrieţi rubricile de des-
10.25 criere a datelor pentru structura din fig. 10.11.

DOCUMENT		COD PRODUS	DEN PRODUS	UM	CANT	PRET
NR	DATA					
n	n	n	a	a-n	n	n
5	6	7	30	3	5+2	7+2

Fig. 10.11.

R. 01 ARTICOL.

```
        02 DOCUMENT
           03 NR              PIC 9(5).
           03 DATA-DOC        PIC 9(6).
        02 COD-PRODUS         PIC 9(7).
        02 DEN-PRODUS         PIC X(30).
        02 UM                 PIC X(3).
        02 CANT               PIC 9(5)V99.
        02 PRET               PIC 9(7)V99.
```

Răspunsul este corect?

DA → 10.26.
NU → 10.15.

10.26 Avînd în vedere că datele pot fi de natură:

— numerică
 → de calcul
 → de editare

— alfanumerică
— alfabetică

prin caracterele utilizate în șabloane trebuie să se facă această distincție.

10.27 *Î*. Menționați caracterele de bază folosite în șabloane pentru descrierea cîmpurilor de date:
— numerice de calcul . . . .
— numerice de editare . . . .
— alfanumerice . . . .
— alfabetice . . . .

R. 9, v; z, 9,., B; X, B; A, B

Pentru descrierea datelor alfabetice se pot utiliza în șabloane următoarele caractere:

— A, pentru a indica poziția unui caracter alfabetic;

— B, pentru a se indica poziția unde se va însera un spațiu (se utilizează la editarea unor date).

Exemple:

```
02 NUME-ȘI-PRENUME     A(30).
03 NUME-ȘI-PRENUME     A(15)BA(15).
```

10.28 Pentru descrierea datelor numerice de calcul se utilizează în șablon caracterele:

9 pentru a indica poziția unei cifre;

V pentru a indica poziția virgulei zecimale (compilatorul nu rezervă spațiu în memoria centrală);

```
02 CANTITATE           PIC 9(7)V99.
```

P pentru a indica poziția unei cifre căreia nu i se rezervă loc în memorie și este interpretată a fi 0;

Exemplu: fie cîmpul PRET descris cu șablonul 999PPP căruia i se atribuie valoarea 245; în calcul se consideră valoarea 245000;

S pentru a indica prezența semnului + sau – (pentru reprezentarea algebrică).

Observație: Acest caracter poate să apară numai în poziția cea mai din stînga a șablonului (prin urmare pentru el nu se rezervă o locație de memorie):

```
02  SOLD      PIC S9(5)V99.
```

10.29 Pentru descrierea datelor numerice de editare se utilizează în șablon caracterele :

9, P, S, Z, +, –, ., ,,CR, DB, *

unde :

Z, indică poziția unei cifre și are ca efect suprimarea zerourilor nesemnificative din față, la cîmpurile ale căror valori se editează, și înlocuirea lor cu spații.

9(5)V99	9(5).99	Z(4)9.99	Z(5).99	Z(5).ZZ
0003455	00034.55	34.55	34.55	34.55
0000000	00000.00	0.00	.00	.
0000032	00000.32	0.32	.32	.32

Caracterul Z nu poate apare la dreapta unui 9 sau a unui *

Exemple eronate

9(4)ZZ

***Z99

10.30 *Caracterul* * indică poziția unei cifre și are ca efect suprimarea zerourilor nesemnificative din fața numărului și înlocuirea lor cu asteriscuri.

9(5)V99	**999.99	9(8)	*(4)99.99
00012.34	**012.34	00127810	**1278.10
00351.45	**351.99	12345678	123456.78

10.31 *Caracterul* + (înserare fixă) indică poziția în care va fi înserat caracterul + dacă valoarea dată este un număr pozitiv sau caracterul –, dacă valoarea dată este un număr negativ:

S9(6)	9(6)+
$22568\bar{2}$	225683–

10.32 *Caracterul* – (înserare fixă) indică poziția în care va fi înserat caracterul ,,spațiu'' dacă valoarea datei este un număr pozitiv, sau caracterul – dacă valoarea datei este un număr negativ. :

S9(6)	9(6)–
$256\bar{4}$	2564–
$235\overset{+}{1}$	2351

10.33 *Caracterul* + (înserare flotantă) permite înserarea unui spațiu în fiecare poziție căreia îi corespunde un zero nesemnificativ și a semnului + sau — în fața primei cifre semnificative în raport cu semnul numărului:

S9(4)V99		+++++9.99
1 | 2 | 7 | 8 (+)		+ | 1 | 2 | . | 7 | 8
0 | 1 | 8 | 3 | 2	++++.999	+ | 1 | . | 8 | 3 | 2

10.34 *Caracterul* — (înserare flotantă) indică înserarea unui spațiu în fiecare poziție căreia îi corespunde un zero nesemnificativ și a spațiului sau a semnului — în fața primei cifre semnificative în raport cu semnul numărului:

9(6)V99	—9,999.99
1999 27	—1,999.27
+1 00	—1.00

10.35 *Î.* Completați tabelul 10.1.

	VALOARE	ȘABLON	VALOARE EDITATĂ
1	1	99.9	
2	2 6 7	****.**	
3	1 4 5	zzz.99	
4	0 0 0 0	zzzz.99	
5	0 0 9 5 2 8 5	*****.**	
6	0 1 9 9 9 2 7	-,---9.99	
7	0 2 7 8 3 0	------9.99	

R. (tabelul 10.2.)

1			0 1 . 0
2			* * * * 2 . 6 7
3			1 4 5 . 0 0
4			. 0 0
5			* * 9 5 2 . 8 5
6			- 1 , 9 9 9 . 2 7
7	2 7 8 3 0	------- .99	- 0 2 7 8 3 0 . 0 0

10.36 *Î.* Completați tabelul 10.3.

DATE	CLASA	ȘABLON
P O P E S C U		
1 9 7 9		
1 3 . 0 3 7 9		
5 0 1 3 2 5		
2 5 3 . 4 3		
7 3 4 X E R O X		

R. (tabelul 10.4)

DATE	CLASA	ŞABLON
POPESCU	Alfabetică	A (10)
1979	Numerică	9999
13 03 79	Numerică de editare	99B99B99
501325	Numerică de calcul	9(5)V99
253 43	Numerică de editare	z(4)9.99
734XEROX	Alfanumerică	x(8)

Răspunsurile date sînt corecte?

DA → 10.37
NU ↶ 10.27

10.37 Un şablon nu poate conţine decît mximum 30 simboluri

PIC XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

trebuie scris

PIC X(40)

10.38 *Clauza USAGE*, se foloseşte pentru a preciza modul de reprezentare a datelor în memorie; ea cuprinde următoarele opţiuni:

```
          ⎧ DISPLAY                       ⎫
          ⎪ ⎧ COMPUTATIONAL—3 ⎫           ⎪
          ⎪ ⎩ COMP—3          ⎭           ⎪
USAGE IS  ⎨ ⎧ COMPUTATIONAL—1 ⎫           ⎬
          ⎪ ⎩ COMP—1          ⎭           ⎪
          ⎪ ⎧ COMPUTATIONAL—2 ⎫           ⎪
          ⎪ ⎩ COMP—2          ⎭           ⎪
          ⎪ ⎧ COMPUTATIONAL   ⎫           ⎪
          ⎩ ⎩ COMP            ⎭           ⎭
```

10.39 *Opţiunea DISPLAY* (implicită) indică formatul de înregistrare zecimal despachetat:

Exemple:

- număr 343
- reprezentare DISPLAY | F | 3 | F | 4 | C | 3 |
- număr —343
- reprezentare DISPLAY | F | 3 | F | 4 | D | 3 |

Observaţie: Lungimea numărului de reprezentat trebuie să fie de maxim 18 octeţi.

10.40 *Opţiunea COMP—3* indică forma de reprezentare împachetată (condensată) a numerelor. Această opţiune se utilizează la descrierea datelor

numerice de calcul înregistrate pe suporturi magnetice (inclusiv memoria centrală) :

Exemple:

- număr 343
- reprezentare COMP—3 | 0 , 0 | 3 , 4 | 3 , C |
- număr —343
- reprezentare COMP—3 | 0 , 0 | 3 , 4 | 3 , D |

Lungimea maximă a zonei de memorie asociată unei date reprezentate în formă zecimal împachetată este de 10 octeți (locații).

10.41 *Opțiunea COMP* indică memorarea datelor în formă binară (se utilizează pentru date ce reprezintă numere întregi pozitive sau negative) :

Exemplu:

SOLD S9(5) 723

| 0 | 2 | D | 3 |

Avantajul acestui format constă în economisirea spațiului necesar pentru memorarea unor date numerice și reducerea timpului de efectuare a operațiilor aritmetice.

Valorile datelor reprezentate în binar (opțiunea COMPUTATIONAL din clauza USAGE) sînt înregistrate în zone de memorie de 2, 4 și respectiv 8 locații în funcție de lungimea în caractere a numărului, după corespondența :

— de la 1 la 4 caractere — 2 locații ;
— de la 5 la 9 caractere — 4 locații ;
— de la 10 la 18 caractere — 8 locații.

10.42 Formatele COMP—1 și COMP—2 sînt mai puțin folosite și indică reprezentarea în virgulă mobilă în simplă precizie și respectiv în dublă precizie.

10.43 *I.* Enumerați opțiunile folosite în cadrul clauzei USAGE pentru a indica forma de reprezentare a datelor

. . . .
. . . .
. . . .
. . . .
. . . .

R. DISPLAY
COMP—3
COMP
COMP—1
COMP—2

10.44 Î. Completaţi în cadrul tabelului 10.5 conţinutul cîmpurilor (în hexa-zecimal) corespunzător şabloanelor folosite în descriere (folosiţi anexa 2).

ŞABLON	FORMAT DE REPREZENTARE	VALOARE	CONŢINUTUL LOCAŢIEI DE MEMORIE
X(5)	DISPLAY	COBOL	
X(6)	DISPLAY	COBOL	
9(5)	DISPLAY	235	
S9(5)	DISPLAY	-20,35	
9(6)	COMP-3	2257	
S9(3)	COMP	723	

R. (tabelul 10.6)

ŞABLON	FORMAT DE REPREZENTARE	VALOARE	CONŢINUTUL LOCAŢIEI DE MEMORIE
X(5)	DISPLAY	COBOL	C3D6C2D6D3
X(6)	DISPLAY	COBOL	C3D6C2D6D3
9(5)	DISPLAY	235	F2F3C5
S9(5)	DISPLAY	-20.35	F2F0F3D2
9(6)	COMP-3	2257	02257C
S9(3)	COMP	723	02D3

10.45 Î. Dacă se omite clauza USAGE din program compilatorul consideră implicită opţiunea . . . . care este singura posibilă pentru date memorate pe suporturi . . . .

R. DISPLAY
nereutilizabile

Răspunsurile sînt corecte?

DA → 10.46
NU ↰ 10.38

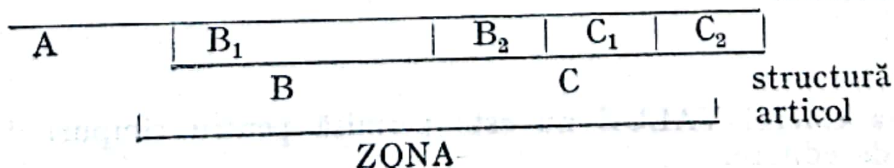
10.46 *Clauza VALUE,* se utilizează în program în secţiunea WORKING–STORAGE (asociată numărului de nivele 88 ea se poate folosi şi-n secţiunea FILE) pentru iniţializarea datelor elementare.

Exemplu:

77 NR–CURENT PIC 9(6) VALUE 0.

10.47 Î. Fie imaginea:

A 3451 FELIX C–256 3456 2343 2563 valori date elementare



Completaţi descrierea desemnînd conţinutul fiecărui cîmp (pentru cîmpurile elementare):

R.
```
01 ZONA.
   02 A   PIC (5)   VALUE 'A3451'.
   02 B.
      03 B1      PIC X(11)      VALUE 'FELIX C—256'.
      03 B2      PIC X(4)       VALUE '3456'.
   02 C.
      03 C1      PIC 9999       VALUE 2343.
      03 C2      PIC 9999       VALUE 2563.
```

10.48 *Î.* Rezultă că utilizarea clauzei VALUE este permisă numai pentru descrierea datelor . . . .
în corelaţie cu clauza . . . .
cu excepţia . . . .

R. — elementare
— PICTURE
— atribuirii de valori unor nume de condiţie în secţiunea FILE sau WORKING—STORAGE.

10.49 Clauza VALUE se mai poate utiliza în cadrul rubricilor de descriere a datelor tip nume—de—condiţie.

Exemplu:
```
01 CARTELA.
   02 COD—CARTELA       PIC 9.
      88 INTRĂRI VALUE 1.
      88 IEŞIRI VALUE 2.
   02 COD—PRODUS        PIC 9(7).
   02 CANTITATE         PIC 9(5)99.
```

10.50 *Î.* Cînd în program se cere iniţializarea unor cîmpuri definite în secţiunea WORKING—STORAGE se foloseşte clauza . . . .

R. VALUE

10.51 Iniţializarea unor cîmpuri, se poate realiza şi cu ajutorul constantelor figurative.

Exemplu:
```
77 TOTAL PIC 9(6)V99        VALUE ZEROS.
01 RÎND.
   02 FILLER PIC X(10)     VALUE SPACES.
   02 NRC PIC ZZZ9.
   02 FILLER PIC X(5)      VALUE ALL 'X'.
   .
   .
   .
```

10.52 Folosirea clauzei VALUE nu este permisă pentru cîmpuri declarate cu şablon de editare.

Exemplu:

eronat:

```
77 T—CANTITATE PIC Z(4)9.99 VALUE 0.
```

corect:

```
77 T—CANTITATE PIC 9(5)V99 VALUE 0.
```

10.53 Inițializarea unui cîmp permite de asemenea introducerea unor constante de prelucrare :

```
77 COEFICIENT—REPARTIZARE PIC 9 VALUE 7.
```

Ați răspuns corect la întrebări ?

DA → Testul 3(pag.355)
NU → Lecția 10



LECȚIA a 11-a SORTAREA FIȘIERELOR PRIN PROGRAM COBOL

- algoritmul de realizare a operației de sortare
- fraza SELECT pentru fișierul de manevră
- rubrica SD
- instrucțiunea SORT
- procedura de intrare
    - instrucțiunea RELEASE
- procedura de ieșire
    - instrucțiunea RETURN

11.1 Se consideră un fișier al intrărilor de valori materiale (tabelul 11.1).

11.2 Articolele se pot grupa în funcție de valorile luate de cîmpurile GESTIUNE și COD—MATERIAL.

11.3 Pentru a obține diverse informații la nivel grupă COD—MATERIAL * și respectiv grupă GESTIUNE, se impune ordonarea articolelor în așa fel încît toate articolele care se referă la același material și respectiv aceeași gestiune să se afle în dispunere succesivă (unul după altul: (tabelul 11.2).

Tabelul 11.1

Gestiune	Cod-material	Preț unitar	Cantitate unitară
02	256781	1020	1000
02	267582	1040	2000
03	289435	1515	1600
02	256781	1020	4000
05	273741	1500	3500
05	293541	3000	4600
03	256594	2020	1500
03	235694	2020	2400
02	235694	2020	5300
02	289431	1515	2700
02	235694	2020	3400

11.4 Operația de ordonare a articolelor dintr-un fișier în funcție de valorile crescătoare sau descrescătoare ale unor cîmpuri numite și chei de sortare (care fac parte din articol) se numește sortare.

Ea se poate realiza:

— *mecanografic*, utilizînd mașini de sortare;

— *electronic*, utilizînd calculatoarele electronice sau mai concret, instrucțiunile specifice pentru realizarea automată a operațiilor de sortare:

---

* Articolele care au aceeași valoare pentru cîmpul COD—MATERIAL formează o grupă În mod asemănător se organizează grupe pentru cîmpul GESTIUNE.

— prin program COBOL;
— prin program utilitar.

11.5 Î. Prin sortare înțelegem deci . . . .

R. operația de ordonare a articolelor într-un fișier în funcție de valorile crescătoare sau descrescătoare al unor cîmpuri numite și chei de sortare.

11.6 Pentru a simplifica scrierea programului de sortare s-au conceput module distincte (incluse în biblioteca sistemului). Programatorul furnizează prin programul COBOL informații pentru a genera instrucțiunile de apel a acestor module.

Editorul de legături reunește aceste module în segmente și le include în textul programului.

Tabelul 11.2

02	235694	2020	3400
02	235694	2020	5300
02	256781	1020	4000
02	256781	1020	1000
02	267582	1040	2000
02	289431	1515	2700
03	235694	2020	2400
03	256594	2020	1500
03	289435	1515	1600
05	273741	1500	3500
05	293541	3000	4600

11.7 Procesul de sortare se derulează în următoarele etape:

a) transferul articolelor din fișierul de intrare în memoria centrală, constituirea monotoniilor * și înregistrarea acestora în fișierul de manevră;

b) interclasarea monotoniilor astfel create;

c) transferul articolelor care constituie monotonia finală într-un fișier de ieșire.

În cadrul programelor de sortare se pot folosi maxim 12 chei pentru care suma lungimilor nu trebuie să depășească 256 octați (fiecare cheie trebuie să aibă lungime fixă și aceeași poziție în cadrul articolelor).

Datele folosite în operațiile de sortare pot fi descrise cu opțiunile de memorare zecimal despachetată, zecimal împachetată, binară sau virgulă flotantă.

11.8 Î. Rezultă că pentru realizarea operațiilor de sortare sînt necesare în general trei fișiere și anume:

. . . .
. . . .
. . . .

R. — fișierul de intrare
— fișierul de ieșire
— fișierul de manevră

11.9 Î. Sortarea se poate face în funcție de una sau maxim . . . . a căror lungime nu trebuie să depășească . . . .

* O grupă ordonată de articole mai este numită și monotonie.

Fiecare cheie trebuie să ocupe . . . .
și să aibă o lungime . . . .
R. 12 chei; 256 octeți; aceeași poziție în cadrul articolului; fixă.

11.10 *În diviziunea ENVIRONMENT* pentru descrierea caracteristicilor fișierului de manevră se utilizează fraza SELECT.

Exemple:

```
SELECT F-MANEVRA ASSIGN BZP.
SELECT F-MANEVRA ASSIGN BAD.
```

Format

SELECT fișier—manevră **ASSIGN** TO { idex—1 | idex—2 | **SYSMAN** }

unde:

— *idex—2, o literă* urmată de codul: { MT | RD | MD | ZP | AD }

— *ZP*, semnifică folosirea unei zone partajate, caz în care fișierul are organizare secvențială—înlănțuită.

11.11 *În diviziunea DATA* descrierea fișierului de manevră în secțiunea FILE se realizează cu ajutorul rubricii SD (Sort—file—Description).

Exemplu:

```
SD  F-MANEVRA
          DATA RECORD ARTICOL-F-MANEVRA.
```

Format general:

SD nume—fișier—manevră

DATA { **RECORD** IS | **RECORDS** ARE } nume—articol . . . .

11.12 *I.* Rubrica SD cuprinde . . . .
R. numele fișierului de manevră și clauza DATA

11.13 Descrierea articolului (articolelor) se realizează cu ajutorul rubricilor de descriere cunoscute din lecția 5.

11.14 **Exemplu** de descriere completă a unui fișier de manevră:

```
SD   F-MANEVRA
     DATA RECORD ARTICOL.
01 ARTICOL.
   02 GESTIUNE-CHEIA-1            PIC 99.
   02 COD-MATERIAL-CHEIA-2        PIC 9(7).
   02 FILLER                      PIC X(14).
```

11.15 *I.* Pentru cîmpurile nesemnificative se utilizează ca nume de dată cuvîntul rezervat . . . .

R. FILLER

11.16 *În diviziunea PROCEDURE* instrucţiunea SORT permite apelul modulelor de sortare din biblioteca sistemului*); ea furnizează informaţiile necesare pentru realizarea sortării şi anume:

— fişierul utilizat în intrare;
— fişierul utilizat în ieşire;
— fişierul de manevră;
— cheile de sortare;

iar dacă este cazul şi procedurile speciale de prelucrare a datelor utilizate în intrare şi respectiv a rezultatelor obţinute prin sortare.

11.17 **Exemplu:**

```
SORT  F-MANEVRA   ASCENDING   GESTIUNE-CHEIE-1
                              COD-MATERIAL-CHEIE-2
                  USING       FISIER-INTRARE
                  GIVING      FISIER-IESIRE.
```

unde:

- F—MANEVRA, numele fişierului de manevră;
- FISIER—INTRARE, numele fişierului utilizat în intrare;
- FISIER—IESIRE, numele fişierului utilizat în ieşire;
- ASCENDING, opţiune pentru a indica sortarea ascendentă;
- GESTIUNE—CHEIE—1, COD—MATERIAL—CHEIE—2, cheile după care se realizează sortarea.

11.18 *Formatul simplificat al instrucţiunii SORT* este:

SORT nume—fişier—manevră { ON { **ASCENDING** / **DESCENDING** } KEY

nume—dată—1 [nume—dată—2] ... } ...

USING nume—fişier—intrare
GIVING nume—fişier—ieşire

unde:

- *Opţiunea ASCENDING* precizează că operaţia de sortare se realizează în funcţie de valorile crescătoare ale cheii (cheilor) de sortare;
- *Opţiunea DESCENDING* precizează că operaţia de sortare se realizează în funcţie de valorile descrescătoare ale cheii (cheilor) de sortare.

Principiul de execuţie al instrucţiunii SORT în format cu opţiunile USING, GIVING este:

- deschiderea fişierelor precizate prin opţiunile USING (fişierul de intrare), GIVING (fişierul de ieşire) şi a fişierului de manevră;
- citirea secvenţială a articolelor fişierului de intrare, constituirea monotoniilor şi memorarea acestora în fişierul de manevră;
- interclasarea monotoniilor pînă la obţinerea monotoniei finale, identică cu fişierul de ieşire;
- citirea secvenţială a articolelor din monotonia finală şi înregistrarea acestora în fişierul de ieşire (precizat prin opţiunea GIVING).

11.19 *Observaţii*:

— cînd sortarea se face după mai multe chei prin program trebuie să se stabilească ierarhia acestora;

---

*) Biblioteca de Subprograme Standard (BSS)

— în clauza KEY din cadrul instrucţiunii SORT sînt specificate cheile de sortare prin numele alese pentru descrierea lor în fişierul de manevră;

— fişierele utilizate în intrare şi respectiv în ieşire trebuie să aibă organizare secvenţială şi să fie memorate pe un suport magnetic; este interzisă deschiderea şi respectiv închiderea acestora prin instrucţiuni COBOL (OPEN, CLOSE).

11.20 Sensul de realizare al operaţiilor este redat în fig. 11.1

11.21 **Exemplu** complet privind diviziunea de procedură pentru realizarea numai a operaţiei de sortare:

```
PROCEDURE DIVISION.
ÎNCEPUT.
    SORT  F-MANEVRA     ASCENDING   GESTIUNE-CHEIE-1
                                    COD-MATERIAL-CHEIE-2
                        USING   FISIER-INTRARE
                        GIVING  FISIER-IESIRE
    STOP RUN.
```

Observaţie Sistemul deschide şi închide în mod automat fişierele prevăzute prin opţiunile GIVING şi USING.

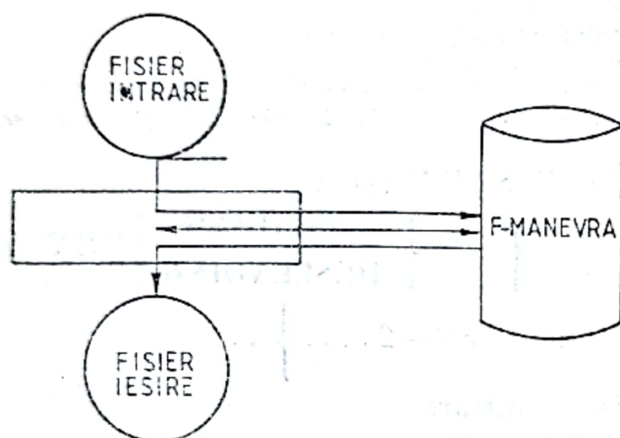
11.22 

Fig. 11.1.

Î. Pentru realizarea operaţiilor de sortare în diviziunea PROCEDURE se utilizează instrucţiunea . . . . care furnizează informaţii în legătură cu . . . .

R. SORT: fişierul utilizat în intrare, fişierul utilizat în ieşire, fişierul de manevră şi cheile de sortare.

11.23 *Î.* Fişierele menţionate în opţiunile: . . . . sînt deschise şi respectiv închise în mod automat de către sistem.

R. USING, GIVING precum şi fişierul de manevră.

11.24 *Î.* Pentru realizarea sortării în ordine crescătoare a articolelor dintr-un fişier se foloseşte opţiunea:

1 ASCENDING
2 DESCENDING

R. 1. ASCENDING

11.25 *Î.* În condiţiile realizării operaţiei de sortare prin program COBOL, trebuie menţionate elementele specifice în diviziunile . . . .

R. ENVIRONMENT, DATA, PROCEDURE

Răspunsurile dumneavoastră sînt corecte?

DA → 11.26
NU ↰ 11.8

11.26 *Procedurile de intrare şi ieşire*

Formatul general al instrucţiunii SORT permite utilizarea unor proceduri de constituire a fişierului de manevră (INPUT PROCEDURE) şi

respectiv de prelucrare a articolelor fişierului de manevră (OUTPUT PROCEDURE).

11.27 **Exemplu:** Controlul datelor dintr-un fişier pe cartele şi ordonarea lor în aşa fel încît fişierul obţinut în ieşire să conţină articolele corecte, ordonate în funcţie de cheile de sortare menţionate (fig. 11.2).

Operaţiile ce se realizează sînt (programul 11.1 pag. 170):

— *controlul datelor* din fişierul FISIER—INTRARE (în schemă FISIER—CARTELE) pentru a vedea dacă sînt corecte; toate articolele sînt afişate în lista de control (cele eronate sînt însoţite de explicaţii referitoare la erorile detectate);

— *furnizarea articolelor* către modulele de sortare;

— *obţinerea în ieşire* a fişierului FISIER—IEŞIRE sortat.

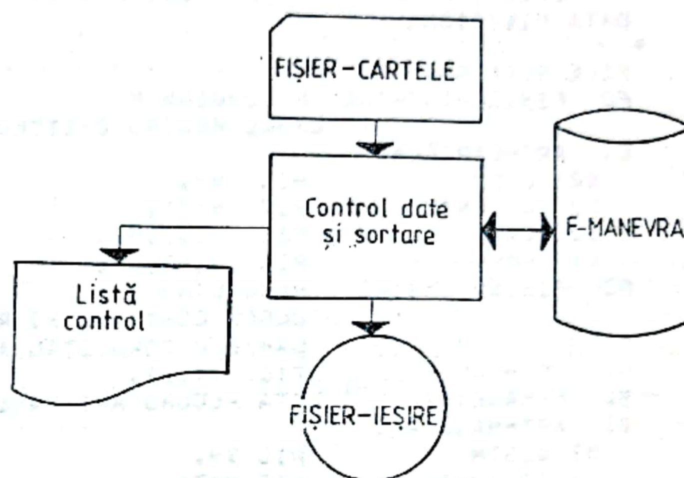


Fig. 11.2.

11.28 *Î.* Acest caz se particularizează prin faptul că . . . .

R. alături de operaţia de sortare apar şi alte operaţii de prelucrare (executate înainte de realizarea sortării).

11.29 Operaţiile de realizat înainte de a începe sortarea sînt grupate într-o procedură distinctă indicată în instrucţiunea SORT astfel:

```
SORT F—MANEVRA ASCENDING COD—MATERIAL—CHEIE
                 INPUT PROCEDURE PROCEDURA—INTRARE
                 GIVING FISIER—IEŞIRE
```

11.30 *Formatul general al instrucţiunii SORT* este:

```
SORT        nume—fişier—manevră

ON { ASCENDING  } KEY nume—cheie—1 [nume—cheie—2 ] ... } ...
   { DESCENDING }

{ USING          nume—fişier—intrare              }
{ INPUT          PROCEDURE IS  nume—secţiune—1    }

{ GIVING         nume—fişier—ieşire              }
{ OUTPUT         PROCEDURE IS nume—secţiune—2    }
```

Observaţie Opţiunea THRU se poate folosi atît în cazul INPUT—PROCEDURE cît şi OUTPUT PROCEDURE cînd procedura folosită la intrare sau ieşire se împarte în secţiuni (sau paragrafe) Ex: **INPUT PROCEDURE** nume—secţiune—1 [**THRU** nume—secţiune—2-

Principiul de execuţie al instrucţiunii SORT cu opţiunile INPUT PROCEDURE, OUTPUT PROCEDURE este:

- se deschide fişierul de manevră;
- se execută instrucţiunile procedurii indicate prin intermediul opţiunii INPUT PROCEDURE;

```
ID DIVISION.
*
PROGRAM-ID.
    SORT01.
*   IN ACEST PROGRAM SE FOLOSESTE INSTRUCTIUNEA SORT IN
*   VARIANTA CU PROCEDURA DE INTRARE
*
ENVIRONMENT DIVISION.
*
CONFIGURATION SECTION.
INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT FISIER-INTRARE ASSIGN SYSIN.
    SELECT FISIER-IESIRE  ASSIGN AMT.
    SELECT F-MANEVRA      ASSIGN BAD.
DATA DIVISION.
*
FILE SECTION.
FD  FISIER-INTRARE  RECORDING F
                   LABEL RECORD OMITTED.
01  ART-CARTELA.
  02  GEST            PIC  99.
  02  COD-MAT         PIC  9(7).
  02  CANT            PIC  9(5)V99.
  02  PRET            PIC  9(4)V99.
FD  FISIER-IESIRE   RECORDING F
                    BLOCK CONTAINS 30 RECORDS
                    LABEL RECORD STANDARD.
01  ART-BANDA       PIC  X(22).
SD  F-MANEVRA      DATA RECORD ART-MANEVRA.
01  ART-MANEVRA.
  02 GESTM            PIC 99.
  02 COD-MATM         PIC 9(7).
  02 FILLER           PIC X(13).
WORKING-STORAGE SECTION.
77  V                 PIC 9 VALUE 0.
777  V-EOF            PIC 9 VALUE 0.
77  C-AE              PIC  9(4) VALUE 0.
PROCEDURE DIVISION.
*
SORTARE SECTION.
INCEPUT.
    SORT F-MANEVRA ASCENDING GESTM
                               COD-MATM
                    INPUT PROCEDURE PROCEDURA-INTRARE
                    GIVING FISIER-IESIRE.
    DISPLAY ' IN F-MANEVRA S-AU INREGISTRAT= ' C-AE 'ARTICOLE'
    STOP RUN.
PROCEDURA-INTRARE SECTION.
MODUL-PRINCIPAL.
    OPEN INPUT FISIER-INTRARE
    READ FISIER-INTRARE AT END MOVE 1 TO V-EOF.
    PERFORM VALIDARE-INREG UNTIL V-EOF = 1
    CLOSE FISIER-INTRARE
    GO TO PARAGRAF-EXIT.
VALIDARE-INREG.
    IF GEST  = 0  OR  GEST > 15
                    MOVE  1  TO  V
                    DISPLAY ' GESTIUNE VALOARE ERONATA '.
    IF CANT  = 0  OR  PRET  = 0
                    MOVE  1  TO  V
                    DISPLAY 'CANTITATE SAU PRET VAL.ER.'.
    IF V = 0  MOVE ART-CARTELA TO ART-MANEVRA
              RELEASE ART-MANEVRA
              ADD 1 TO C-AE
        ELSE  DISPLAY ' ARTICOL ERONAT = ' ART-CARTELA.
    READ FISIER-INTRARE AT END MOVE 1 TO V-EOF.
PARAGRAF-EXIT.
    EXIT.
```

Programul 11.1.

- se interclasează monotoniile pînă se ajunge la monotonia finală;
- se execută instrucțiunile procedurii indicate prin intermediul opțiunii OUTPUT PROCEDURE;
- se închide fișierul de manevră

11.31 *Î.* Din cele de mai sus rezultă că pentru realizarea operațiilor de sortare putem întîlni următoarele cazuri:

. . . .

R. — sortare simplă
— sortare cu procedură de intrare
— sortare cu procedură de ieșire
— sortare cu proceduri de intrare și ieșire

11.32 Articolele prelucrate cu ajutorul operațiilor prevăzute în secțiunea INPUT PROCEDURE sînt înregistrate pe fișierul de manevră folosind instrucțiunea RELEASE (care este similară instrucțiunii WRITE în formatul folosit pentru înregistrarea articolelor pe fișiere cu organizare secvențială, memorate pe suporturi magnetice).

Format:

RELEASE nume—articol—manevră
[**FROM** nume—dată]

Ați înțeles schema privind fluxul datelor de la paragraful 11.28?

DA → continuați citirea acestui paragraf
NU ↰ 11.28

Din schemă rezultă că procedura de intrare va cuprinde operațiile de validare a articolelor citite din FISIER—CARTELE · operațiile de listare a articolelor în lista de control* și operațiile de furnizare a articolelor, modulelor de sortare (folosind instrucțiunea RELEASE).

11.33 În condițiile în care se cere realizarea unor operații de prelucrare, aplicate fișierului sortat, acestea se grupează într-o procedură distinctă de ieșire.

11.34 **Exemplu:** (fig. 11.3)

Fie F—BANDA un fișier care conține date referitoare la oamenii muncii dintr-o unitate economică. Se dorește ca prin program să se ordoneze articolele din F—BANDA în funcție de cheile LOC — MUNCA, NUMĂR—MARCA, iar articolele ordonate să se listeze (folosind instrucțiunea de afișare DISPLAY) într-o situație centralizatoare pe locuri

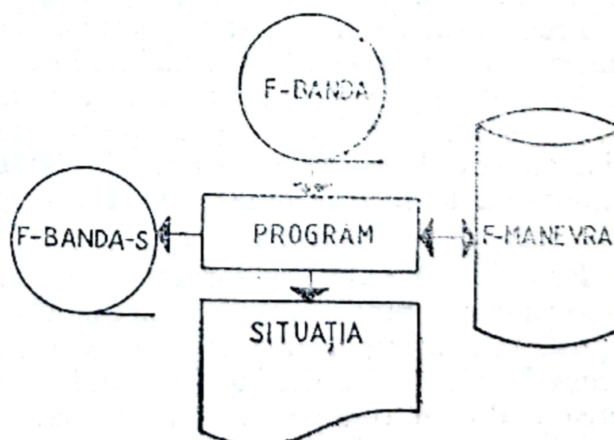


Fig. 11.3.

* Întocmită pentru a controla datele incluse în prelucrare și a semnala erorile detectate.

de muncă și de asemenea să se memoreze într-un fișier pe bandă magnetică numit F—BANDA—S (programul 11.2).

11.35 𝔍. Instrucțiunea SORT are structura

. . . .
. . . .
. . . .

ℛ. SORT F—MANEVRA ASCENDING LOC—MUNCA—M
NUMĂR—MARCA—M
USING F—BANDA
OUTPUT PROCEDURE
PROCEDURA—IEȘIRE

11.36 Procedura de ieșire (vezi program SORTARE) va cuprinde operațiile necesare:

— listării articolelor sortate la imprimantă·

— înregistrării articolelor sortate într-un fișier pe bandă magnetică numit F—BANDA—S (acest fișier va trebui să fie deschis și respectiv închis de către programator).

Pentru a avea acces la articolele sortate (care se găsesc în fișierul de manevră) se utilizează instrucțiunea RETURN.

11.37 **Exemplu:**

```
RETURN F—MANEVRA AT END MOVE 'DA' TO SF—MANEVRA.
```

Instrucțiunea RETURN are structură similară instrucțiunii READ pentru accesul secvențial la date (vezi lecția 6):

RETURN nume—fișier—manevră
[**INTO** nume—dată]
AT **END** instrucțiune ....

11.38 După executarea instrucțiunii SORT se continuă cu instrucțiunea imediat următoare celei de sortare. Programele 11.3 a și 11.3 b constituie două variante de scriere a programelor care utilizează instrucțiunea SORT în opțiunile cu procedură de intrare și respectiv de ieșire.

11.39 Procedurile de intrare și respectiv de ieșire nu trebuie să conțină instrucțiunea GO TO care să facă trimiteri în afara acestora. Procedura de intrare poate conține instrucțiuni de apel a unui paragraf dintr-o altă procedură.

Procedurile de intrare și respectiv de ieșire pot conține instrucțiunea STOP RUN. Folosirea instrucțiunii STOP RUN în aceste secțiuni permite renunțarea la instrucțiunea GO TO pentru a face saltul la un paragraf care desemnează sfîrșitul secțiunii.

Funcțiile de validare, sortare, și editare se pot realiza și fără a proceda la secționarea programului, în această variantă articolele validate vor fi depuse într-un fișier distinct de cel de manevră iar cele sortate de asemenea într-un fișier distinct (de unde sînt în continuare prelucrate în vederea realizării funcției de editare sau a altei funcții).

Pentru verificarea cunoștințelor rezolvați testul 4, pagina 357

```
 ID DIVISION.
*
 PROGRAM-ID.
     SORT02.
**  FLUXUL GENERAL AL DATELOR PENTRU PROGRAMUL  *SORTARE*
**  ESTE URMATORUL:
**                      F-BANDA -> SORTARE <-> F-MANEVRA
**                                    I
**                                    V
**                           ----------------
**                           I                I
**                           V                V
**                     LISTA               F-BANDA-S
**                        DE
**                         CONTROL
**
**     PROCEDURA DE IESIRE LISTEAZA(FOLOSIND INSTRUCTIUNEA
**     DISPLAY) ARTICOLELE DIN F-MANEVRA SI CREAZA FISIERUL
**     F-BANDA-S
**
**   ACEST PROGRAM FOLOSESTE INSTRUCTIUNEA SORT IN VARIANTA
**   CU PROCEDURA IN IESIRE
**   INTRUCIT DIN SECTIUNE MODUL-PRINCIPAL SE FACE APELUL
**   SECTIUNII PROCEDURA-IESIRE S-A ALES VARIANTA PRECIZARII
**   PUNCTULUI DE  IESIRE INTITULAT SFIRSIT-SECTIUNE SI PRIN
**   URMARE FOLOSIRII INSTRUCTIUNII GO TO
**
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT F-BANDA        ASSIGN AMT.
     SELECT F-BANDA-S      ASSIGN BMT.
     SELECT F-MANEVRA      ASSIGN CZP.
 DATA DIVISION.
*
 FILE SECTION.
 FD  F-BANDA            RECORDING F
                        BLOCK CONTAINS 30 RECORDS
                        LABEL RECORD STANDARD.
 01  ARTICOL-BANDA.
   02 LOC-MUNCA         PIC 99.
   02 NUMAR-MARCA       PIC 9(5).
   02 NUME-PRENUM       PIC  X(40).
 FD  F-BANDA-S          RECORDING F
                        BLOCK CONTAINS 30 RECORDS
                        LABEL RECORD STANDARD.
 01  ARTICOL-BAND-S   PIC X(47).
 SD  F-MANEVRA DATA RECORD ART-MANEVRA.
 01  ART-MANEVRA.
   02 LOC-MUNCA-M      PIC 99.
   02 NUMAR-MARCA-M    PIC 9(5).
   02 NUME-PRENUME-M   PIC X(40).
 WORKING-STORAGE SECTION.
 77  IND-SF                PIC XX VALUE 'NU'.
   88 IND-SF-DA                   VALUE 'DA'.
 77  NRC                    PIC 999 VALUE 0.
 PROCEDURE DIVISION.
```

Programul 11.2.

```
*
 MODUL-PRINCIPAL SECTION.
 START.
     SORT F-MANEVRA ON ASCENDING KEY  LOC-MUNCA-M
                                       NUMAR-MARCA-M
                         USING F-BANDA
                         OUTPUT PROCEDURE PROCEDURA-IESIRE
     STOP RUN.
 PROCEDURA-IESIRE SECTION.
 MODUL-LISTARE.
     OPEN OUTPUT F-BANDA-S
     DISPLAY SPACES
     DISPLAY ' LISTA  CENTRALIZATOARE
     DISPLAY SPACES
     RETURN F-MANEVRA AT END MOVE 'DA' TO IND-SF.
     PERFORM LISTARE UNTIL IND-SF-DA
     DISPLAY SPACES
     DISPLAY '========================================
     GO TO SFIRSIT-SECTIUNE.
 LISTARE.
     ADD 1  TO NRC
     DISPLAY ' * ' NRC  ' ' ART-MANEVRA
     WRITE ARTICOL-BAND-S FROM ART-MANEVRA
     RETURN F-MANEVRA AT END MOVE 'DA' TO IND-SF.
 SFIRSIT-SECTIUNE.
     EXIT.
*IN VARIANTA  FARA INSTRUCTIUNEA DE PROCEDURA DIVIZIUNEA DE
*PROCEDURA POATE AVEA STRUCTURA:
*MODUL-PRINCIPAL SECTION.
*START.
*    SORT F-MANEVRA ON ASCENDING KEY LOC-MUNCA-M
*                                    NUMAR-MARCA-M
*                       USING F-BANDA
*                       OUTPUT PROCEDURE PROCEDURA-IESIRE.
*PROCEDURA-IESIRE SECTION.
*MODUL-LISTARE.
*    OPEN OUTPUT F-BANDA-S
*    DISPLAY SPACES
*    DISPLAY ' LISTA CENTRALIZATOARE '
*    DISPLAY SPACES
*    RETURN F-MANEVRA AT END MOVE 'DA' TO IND-SF.
*    PERFORM LISTARE UNTIL IND-SF-DA
*    DISPLAY SPACES
*    DISPLAY '========================================'
*    STOP RUN.
*LISTARE.
*    ADD 1 TO NRC
*    DISPLAY ' * ' NRC ' ' ART-MANEVRA
*    WRITE ARTICOL-BANDA-S FROM ART-MANEVRA
*    RETURN F-MANEVRA AT END MOVE 'DA' TO IND-SF.
*
* DUPA CUM REZULTA DIVIZIUNEA DE PROCEDURA IN VARIANTA
* FARA GO TO A FOST SCRISA SUB FORMA COMENTARIILOR : FOLOSIREA
* EFECTIVA SE POATE REALIZA PRIN ELIMINAREA CARACTERULUI * DIN
* COLOANA 7 A FIECAREI LINII SURSA
*
*
*                        PROGRAMUL 11.2
```

Programa 11.2 continuare

```
 ID DIVISION.
*
 PROGRAM-ID. SORT03.
* ACEST PROGRAM FOLOSESTE INSTRUCTIUNEA SORT IN VARIANTA
* CU PROCEDURI IN INTRARE SI RESPECTIV IESIRE !EL REALIZEAZA
* URMATOARELE FUNCTII:
*       -VALIDEAZA DATELE DIN FISIER-CARTELE
*       -SCRIE ARTICOLELE CORECTE IN FISIER-MANEVRA
*       -SORTEAZA ARTICOLELE FISIERULUI FISIER-MANEVRA
*       -CREAZA FISIERUL F-SECV-IND CU ORGANIZARE
*          SECVENTIALA INDEXATA
*
*
  ENVIRONMENT DIVISION.
*
  CONFIGURATION SECTION.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
     SELECT FISIER-CARTELE  ASSIGN SYSIN.
     SELECT FISIER-MANEVRA  ASSIGN BAD.
     SELECT F-SECV-IND      ASSIGN AAD
                            ORGANIZATION INDEXED
                            RECORD KEY COD-MATD.
 DATA DIVISION.
*
 FILE SECTION.
 FD  FISIER-CARTELE     RECORDING F
                        LABEL RECORD OMITTED.
 01  ART-CARTELA.
   02 DEN-MAT            PIC X(40).
   02 COD-MAT            PIC 9(7).
   02 UM                 PIC XXX.
   02 STOC               PIC 9(5)V99.
   02 PRET               PIC 9(4)V99.
 FD  F-SECV-IND          RECORDING F
                         LABEL RECORD STANDARD.
 01  ART-DISC.
   02 DEN-MATD           PIC X(40).
   02 COD-MATD           PIC 9(7).
   02 UMD                PIC XXX.
   02 STOCD              PIC 9(5)V99.
   02 PRETD              PIC 9(4)V99.
 SD  FISIER-MANEVRA
     DATA RECORD ARTICOL.
 01  ARTICOL.
   02 DEN-MATM           PIC X(40).
   02 CHEIE-SORTARE      PIC 9(7).
   02 UMM                PIC XXX.
   02 STOCM              PIC 9(5)V99.
   02 PRETM              PIC 9(4)V99.
 WORKING-STORAGE SECTION.
 77  V-EOF               PIC 9 VALUE 0.
 77  V-IVK               PIC 9.
 77  V-AI                PIC 9(4) VALUE 0.
 PROCEDURE DIVISION.
```

Programul 11.3. a

```
SORTARE SECTION.
INCEPUT.
    SORT FISIER-MANEVRA ASCENDING CHEIE-SORTARE
              INPUT PROCEDURE INTRARE
             OUTPUT PROCEDURE IESIRE
    STOP RUN.
INTRARE SECTION.
MODUL-PRINCIPAL.
    OPEN INPUT FISIER-CARTELE
    READ FISIER-CARTELE AT END MOVE 1 TO V-EOF.
    PERFORM VALIDARE-INREG UNTIL V-EOF = 1
    CLOSE FISIER-CARTELE
    GO TO PARAGRAF-EXIT1.
VALIDARE-INREG.
    DISPLAY ART-CARTELA
    IF COD-MAT NUMERIC MOVE DEN-MAT TO DEN-MATM
                       MOVE COD-MAT  TO CHEIE-SORTARE
                       MOVE UM       TO UMM
                       MOVE STOC     TO STOCM
                       MOVE PRET     TO PRETM
                       RELEASE ARTICOL
      ELSE DISPLAY 'CARTELA ERONATA = ' ART-CARTELA.
    READ FISIER-CARTELE AT END MOVE 1 TO V-EOF.
PARAGRAF-EXIT1.
    EXIT.
IESIRE SECTION.
MODUL-PRINCIPAL.
    OPEN OUTPUT F-SECV-IND
    MOVE 0 TO V-EOF
    RETURN FISIER-MANEVRA AT END MOVE 1 TO V-EOF.
    PERFORM INDEXARE UNTIL V-EOF = 1
    DISPLAY ' IN FISIERUL F-SECV-IND S-AU INREG. ' V-AI
                                                  ' ARTICOLE
    CLOSE F-SECV-IND
    GO TO PARAGRAF-EXIT2.
INDEXARE.
    MOVE 0 TO V-IVK
    MOVE ARTICOL TO ART-DISC
    WRITE ART-DISC INVALID KEY MOVE 1 TO V-IVK.
    IF V-IVK = 1
            THEN
                DISPLAY ' CHEIE  ERONATA = ' ARTICOL
            ELSE
                ADD 1 TO V-AI.
    RETURN FISIER-MANEVRA AT END MOVE 1 TO V-EOF.
PARAGRAF-EXIT2.
    EXIT.
```

Programul 11.3.a (continuare)

```
.
.
.
PROCEDURE DIVISION.
*
SORTATE SECION.
INCEPUT.
    SORT FISIER—MANEVRA  ASCENDING CHEIE—SORTARE
                         INPUT PROCEDURE INTRARE
                         OUTPUT PROCEDURE IESIRE.
INTRARE SECTION.
MODUL—PRINCIPAL.
    OPEN INPUT FISIER—CARTELE
    READ FISIER—CARTELE AT END MOVE 1 TO V—EOF.
    PERFORM VALIDARE—INREG UNTIL V—EOF = 1
    CLOSE FISIER—CARTELE.
SF.
    EXIT.
IESIRE SECTION.
MODUL—PRINCIPAL2.
    OPEN OUTPUT F—SECV—IND
    MOVE 0 TO V—EOF
    RETURN FISIER—MANEVRA AT END MOVE 1 TO V—EOF.
    PERFORM INDEXARE UNTIL V—EOF = 1
    DISPLAY ' IN FISIERUL F—SECV—IND S—AU INREG. ' V—AI ' ARTI-
    COLE'
    CLOSE F—SECV—IND
    STOP RUN.
INDEXARE.
    MOVE 0 TO V—IVK
    MOVE ARTICOL TO ART—DISC
    WRITE ART—DISC INVALID KEY MOVE 1 TO V—IVK.
    IF V—IVK = 1
        THEN
            DISPLAY 'CHEIE ERONATA =' ARTICOL
        ELSE
            ADD 1 TO V—AI.
    RETURN FISIER—MANEVRA AT END MOVE 1 TO V—EOF.
VALIDARE—INREG.
    DISPLAY ART—CARTELA
    IF COD—MAT NUMERIC
        THEN          .
                      .
        ELSE          .
                      .
                      .
                      .
                      .
    READ FISIER—CARTELE AT END MOVE 1 TO V—EOF.
*   PROGRAM DE SORTARE IN VARIANTA FARA INSTRUCTIUNEA GO TO
```

Programul 11.3. b

LECȚIA a 12-a INSTRUCȚIUNI DE LUCRU CU FIȘIERE MEMORATE PE SUPORTURI MAGNETICE

- alte opțiuni ale instrucțiunilor OPEN, CLOSE
- instrucțiunea READ
  - opțiunea INTO
  - opțiunea INVALID KEY
- instrucțiunea WRITE
  - opțiunea FROM
  - opțiunea INVALID KEY
- instrucțiunea REWRITE
- instrucțiunea DELETE

12.1 *Instrucțiunea OPEN* realizează deschiderea fișierelor, operație în care se includ:

— verificarea disponibilității zonelor de intrare/ieșire alocate fișierelor;

— verificarea disponibilităților perifericelor:

— crearea etichetelor (pentru operații de creare de fișiere);

— verificarea prezenței fișierelor și a etichetelor (pentru operații de consultare de fișiere).

Formatul general al instrucțiunii OPEN este:

```
     ⎧ INPUT { nume—fișier [ ⎧REVERSED          ⎫ ] } ...     ⎫
     ⎪                       ⎩WITH NO REWIND    ⎭              ⎪
OPEN ⎨ OUTPUT {nume—fișier  [WITH NO REWIND]} ...              ⎬
     ⎪   ⎧INPUT—OUTPUT⎫                                        ⎪
     ⎩   ⎩I—O         ⎭   {nume—fișier} ...                    ⎭
```

unde:

- *opțiunea INPUT—OUTPUT*, indică faptul că fișierul memorat pe disc magnetic va fi actualizat (va fi utilizat atît în intrare cît și în ieșire);
- *opțiunea NO REWIND*, indică suprimarea rebobinării benzii (se folosește numai la fișiere memorate pe bandă magnetică);
- *opțiunea REVERSED*, indică citirea înapoi a fișierului și se folosește numai pentru fișiere pe bandă magnetică. În figura 12.1 fișierul F—CART trebuie deschis în intrare iar fișierul F—DISC în intrare—ieșire.

12.2 Fişierele organizate secvenţial pot fi citite începînd cu primul sau cu ultimul articol. Fişierele pe bandă magnetică sau disc magnetic se citesc de obicei de la început către sfîrşit. Din acest motiv la deschiderea fişierelor, rolele de benzi magnetice se rebobinează automat (se revine la începutul benzii). Opţiunea *WITH NO REWIND* se foloseşte pentru suprimarea rebobinării benzii. Citirea înapoi (opţiunea REVERSED) se poate realiza numai la fişierele care au articole de format fix sau nedefinit.

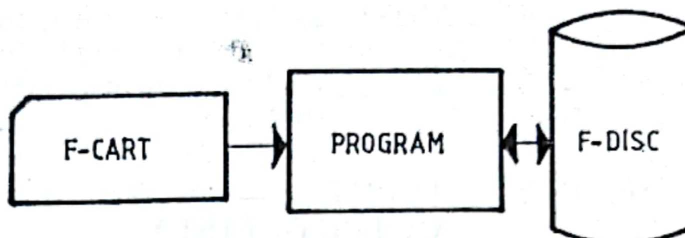


Fig. 12.1.

12.3 *Opţiunea WITH NO REWIND* se recomandă atunci cînd sînt mai multe fişiere pe bandă magnetică. Structurile multifişier pe bandă magnetică se declară prin clauza MULTIPLE FILE din paragraful I—O—CONTROL. Compilatorul construieşte un tabel în care reţine idexul fiecărui fişier şi poziţia acestuia în structura multifişier.

12.4 Într-un program pot fi supuse prelucrării, în mod succesiv, toate datele memorate într-o structură multifişier sau numai o parte din acestea (dintr-un anumit fişier). Parcurgerea secvenţială a fişierelor este obligatorie pentru a asigura poziţionarea capului de citire la fişierul de citit. Prin urmare fişierele ce preced fişierul de supus prelucrării trebuie deschise, citite şi închise cu precizarea suprimării rebobinării benzii.

12.5 **Exemplu:**

Pe o bandă magnetică sînt memorate 3 fişiere: F1, F2, F3 dispuse în această ordine pe suport.

Se cere consultarea fişierului F2. Pentru aceasta trebuie deschis, citit şi închis fişierul F1; la deschiderea fişierului se utilizează opţiunea de anulare a rebobinării.

```
ENVIRONMENT DIVISION.
INPUT—OUTPUT SECTION.
FILE—CONTROL.
        SELECT F1 ASSIGN TO A.
        SELECT F2 ASSIGN TO B.
        SELECT F3 ASSIGN TO C.
I—0—CONTROL.
        SAME AREA FOR F1, F2, F3
        MULTIPLE FILE F1, F2, F3.
DATA DIVISION.
:. . . . . . . . . . . .
PROCEDURA DIVISION.
A.      OPEN INPUT F1
        READ F1 AT END MOVE 1 TO IND—SF.
. . . . . . . . . . . . . . .
        CLOSE F1 WITH NO REWIND
        OPEN INPUT F2 WITH NO REWIND
. . . . . . . . . → operaţii de prelucrare
        CLOSE F2
```

În exemplul de mai sus dispozitivul de citire se poziţionează la începutul fişierului F2 care urmează a fi prelucrat şi închis în mod obişnuit.

12.6 Într-un program COBOL, pentru orice fișier utilizat (cu excepția celor de sortare), operația de deschidere (OPEN) trebuie să preceadă pe cele de acces (citire, scriere).

12.7 *Î.* Fie fișierele din fig. 12.2 unde:
— F—BAND1, fișierul de intrare în instrucțiunea SORT;
— F—BAND2, fișierul de ieșire în instrucțiunea SORT;
— LUCRU, fișierul de manevră în instrucțiunea SORT;
— LISTA, fișierul obținut prin listarea articolelor din F—BAND2;
Instrucțiunea OPEN se scrie în acest caz . . . .
. . . .

R. OPEN INPUT F—BAND2
OUTPUT LISTA

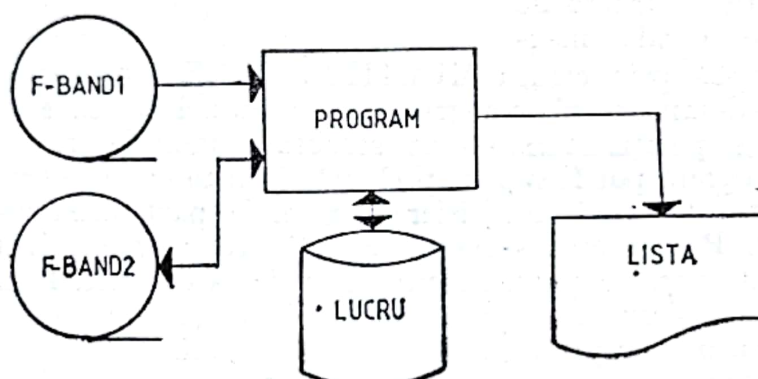


Fig. 12.2.

12.8 *Instrucțiunea CLOSE* descrie operația de închidere a fișierelor, iar în cazul fișierelor memorate pe suporturi magnetice crearea (dacă fișierul este închis după ce a fost creat) și respectiv controlul (dacă fișierul este închis după ce a fost consultat) etichetelor.

Formatul general al instrucțiunii CLOSE este:

$$\textbf{CLOSE}\ \text{nume—fișier}\ \left[\begin{Bmatrix}\textbf{REEL}\\ \textbf{UNIT}\end{Bmatrix}\right]\left[\text{WITH}\begin{Bmatrix}\textbf{NO REWIND}\\ \textbf{LOCK}\end{Bmatrix}\right]\ \ldots$$

12.9 Un fișier se poate afla pe mai multe volume de bandă magnetică (fișier multivolum). La închiderea volumului, a cărui prelucrare s-a terminat, se folosește opțiunea REEL pentru a trece controlul la următorul volum.

În cazul cînd se utilizează un fișier multivolum, pe disc magnetic, se folosește opțiunea UNIT.

12.10 La închiderea fișierelor pe bandă magnetică se execută în mod automat rebobinarea benzii. Pentru anularea rebobinării în vederea citirii unui alt fișier ce se află pe același volum se folosește opțiunea WITH NO REWIND.

Exemplu:

```
CLOSE FIȘIER—BANDA—1 WITH NO REWIND
```

12.11 În mod normal un fișier închis prin instrucțiunea CLOSE rămîne disponibil pentru același program. Dacă se mai utilizează în programul respectiv va fi redeschis cu o instrucțiune OPEN. Prin utilizarea *opțiunii*

WITH LOCK fișierul închis nu mai poate fi redeschis în același program ca urmare a eliberării perifericului.

Exemplu:

CLOSE MISCARI WITH LOCK

12.12 Î. Pentru fișierele utilizate în intrare instrucțiunea CLOSE realizează controlul . . . . (care pot fi standard și utilizator. Așa cum s-a precizat în clauza . . . . din secțiunea . . . . diviziunea . . . . Controlul nu are sens atunci cînd în clauza LABEL s-a precizat opțiunea . . . . Etichetele au fost create atunci cînd fișierele au fost deschise cu opțiunea . . . .

R. — etichetelor
— LABEL
— FILE
— DATA
— OMITTED
— OUTPUT

12.13 Î. Opțiunea REVERSED se utilizează pentru fișierele pe . . . . și precizează citirea . . . . Opțiunea WITH NO REWIND se folosește pentru fișierele pe . . . . și realizează . . . . Opțiunea INPUT—OUTPUT se folosește pentru fișierele pe . . . . . . în cazul operațiunii de . . . .
Opțiunea REEL se utilizează în instrucțiunea . . . . la fișierele . . . . pe suportul . . . . pentru a trece controlul la următorul . . . .
Opțiunea UNIT se folosește la fișierele . . . . pe suportul . . . . iar opțiunea WITH LOCK are ca efect . . . .

R. — bandă magnetică și disc magnetic
— înapoi
— bandă magnetică
— suprimarea rebobinării benzii
— disc magnetic
— actualizare
— CLOSE
— multivolum
— bandă magnetică
— volum
— multivolum
— disc magnetic
— eliberarea perifericului

12.14 La prezentarea instrucțiunii READ într-un format simplificat, adecvat exploatării secvențiale, s-a spus că aceasta realizează transferul unui articol din fișier într-o zonă din memoria centrală (principală) afectată acelui

fişier. Articolul adus în zona articol asociată va fi disponibil pentru prelucrare.

12.15 Anumite prelucrări cer ca articolul să fie prezentat nu numai în zona—articol alocată dar şi într-o altă zonă din memorie; caz în care programatorul trebuie să utilizeze pentru scrierea instrucţiunii READ următorul format:

READ nume—fişier [RECORD] [**INTO** nume—dată]
AT **END** instruc-ţiune—imperativă ....

12.16 Prin *opţiunea INTO* (ce se poate folosi numai cu articole de format fix) articolul se transferă şi în zona indicată prin *nume—dată*. Aceasta poate indica o zonă de lucru definită în WORKING—STORAGE SECTION, sau o zonă ce aparţine unui fişier deschis în OUTPUT (definit în FILE SECTION).

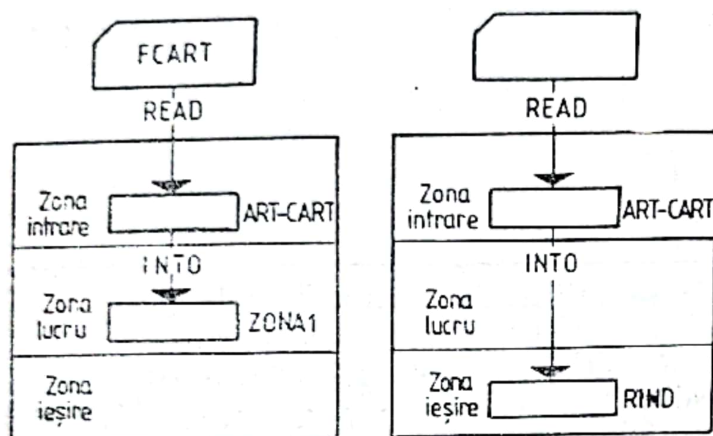


Fig. 12.3.

Exemplu: fig. 12.3.

12.17 În cazul fişierelor memorate pe disc magnetic, cu oganizare secvenţială—indexată sau selectivă, consultate în acces direct, este necesară realizarea unui control pentru a constata dacă citirea are sens (articolul căutat se găseşte în cadrul fişierului).

12.18 Operaţia de consultare în acces direct a unui fişier se realizează cu ajutorul instrucţiunii READ:

READ nume—fişier [**INTO** nume—dată]
INVALID KEY instrucţiune ....

Conform acestui format, dacă articolul este găsit în fişier, execuţia programului continuă cu instrucţiunea imediat următoare de după punct; în caz contrar se trece la executarea instrucţiunii (instrucţiunilor) precizate prin clauza INVALID KEY.

12.19 În fraza SELECT fişierul FIŞIER—DISC (cu organizare secvenţială—indexată) consultat în acces direct apare descris astfel:

```
SELECT  FIŞIER—DISC ASSIGN AAD
        ORGANIZATION INDEXED
        ACCESS RANDOM
        RECORD KEY COD—MATERIAL
        SYMBOLIC KEY CHEIE—ACCES.
```

12.20 Cheia de identificare a articolului (indicată prin clauza RECORD KEY) constituie un cîmp din structura articolului asociat fişierului cu organizare secvenţială—indexată iar cheia de acces (indicată prin clauza SYMBOLIC KEY) constituie un cîmp definit în secţiunea WORKING—STORAGE sau în secţiunea FILE în structura unui alt tip de articol (decît cel asociat fişierului cu organizare secvenţială—indexată).

Exemple:

a)

```
DATA DIVISION.
FILE SECTION.
FD FISIER—DISC    RECORDING F
                  BLOCK CONTAINS 40 RECORDS
                  LABEL RECORD STANDARD.
01   ARTICOL—DISC—SI.
  02 COD—MATERIAL                 PIC 9(8) COMP—3.
  02 DENUMIRE—MATERIAL            PIC X(40).
  02 UNITATE—MĂSURĂ               PIC XXX.
  02 PRET—UNITAR                  PIC 9(5)V99 COMP—3.
  02 STOC                         PIC 9(7)V99 COMP—3.
.
.
.
WORKING—STORAGE SECTION.
77 CHEIE—ACCES                    PIC 9(8) COMP—3.
.
.
.
```

b)

```
DATA DIVISION.
FILE SECTION.
FD FISIER—DISC    RECORDING F
                  BLOCK CONTAINS 50 RECORDS
                  LABEL RECORD STANDARD.
01 ARTICOL—DISC—SI.
  02 COD—MATERIAL                 PIC 9(9) COMP—3.
  02 DENUMIRE—MATERIAL            PIC X(30).
  02 UNITATE—MĂSURĂ               PIC XXX.
  02 PRET—UNITAR                  PIC 9(7)V99 COMP—3.
  02 STOC                         PIC 9(7)V99 COMP—3.
FD FISIER—BANDA   RECORDING F
                  BLOCK CONTAINS 60 RECORDS
                  LABEL RECORD STANDARD.
01 ARTICOL—BANDA.
  02 CHEIE—ACCES                  PIC 9(8) COMP—3.
  02 DOCUMENT.
     03 NUMĂR                     PIC 9(5).
     03 DATA—DOCUMENT             PIC 9(6).
  02 COD—GESTIUNE                 PIC 99.
  02 CANTITATE—INTRATĂ            PIC 9(5)V99 COMP—3.
```

12.21 *Instrucţiunea WRITE*, descrie operaţia de înregistrare (scriere) în fişierul deschis în ieşire (cu opţiunea OUTPUT din instrucţiunea OPEN). Pentru fişierele pe bandă magnetică se utilizează formatul:

a) **WRITE** nume—articol

unde:

- *nume—articol* numele tipului de articol asociat fişierului.

12.22 Atunci cînd articolele fişierului sînt blocate, la o instrucţiune de scriere, articolul se depune în zona tampon iar scrierea efectivă în banda magnetică are loc după completarea blocului.

12.23 După scrierea articolului în fişier, conţinutul zonei articol nu mai este disponibil (idexul fişierulul pe bandă este D).

Exemplu:
```
ENVIRONMENT DIVISION.
INPUT—OUTPUT SECTION.
FILE—CONTROL.
    SELECT FBAND ASSIGN TO D.
    . . . . . . . . . . . . . . . . . . . . .
    . . . . . . . . . . . . . . . . . . . . .
DATA DIVISION.
FILE SECTION.
FD FBAND . . . . . . . . . . . . . .
    . . . . . . . . . . . . . . . . . .
01 ART—BAND.
    02 . . . . . . . . . . . . . . . . . . .
    . . . . . . . . . . . . . . . . . . . .
    02 . . . . . . . . . . . . . . . . . . .
PROCEDURE DIVISION.
ÎNCEPUT.
    OPEN INPUT . . . . . . . . . . . .OUTPUT FBAND
    . . . . . . . . . . . . . . . . . . . .
    . . . . . . . . . . . . . . . . . . . .
    WRITE ART—BAND
    . . . . . . . . . . . . . . . . . . . .
```

12.25 *Î.* Cu ce instrucțiune au fost aduse datele din alte zone ale memoriei în zona ART—BAND în vederea scrierii lui?

. . . .

R. MOVE

12.26 Dacă structura articolului care se scrie este identică cu structura articolului care se citește este rațional ca transferul să nu se facă element cu element ci în totalitate.

12.27 *Instrucțiunea WRITE* cu opțiunea FROM determină transferul datelor din *nume—dată* în *nume—articol* nemaifiind necesară utilizarea instrucțiunii MOVE:

WRITE nume—articol [**FROM** nume—dată]

Opțiunea FROM se utilizează atunci cînd *nume—articol* și *nume—dată* au structuri identice.

Exemplu:
```
DATA DIVISION.
FILE SECTION.
FD FISIER—INTRARE RECORDING F
                  BLOCK CONTAINS 30 RECORDS
                  LABEL RECORD STANDARD.
01 ARTICOL—INTRARE.
  02 DATA—11                     PIC X(15).
  02 DATA—12                     PIC X(25).
  02 DATA—13                     PIC 9(7)V99.
FD FISIER—IESIRE  RECORDING F
                  BLOCK CONTAINS 35 RECORDS
                  LABEL—RECORD STANDARD.
01 ARTICOL—IESIRE.
  02 DATA—21                     PIC X(15).
  02 DATA—22                     PIC X(25).
  02 DATA—23                     PIC 9(7)V99.
.
.
.
```

```
PROCEDURE DIVISION.
    .
    .
    .
    READ FISIER-INTRARE
                   AT END MOVE 1 TO V-EOF.
    .
    .
    .
    WRITE ARTICOL-IESIRE FROM ARTICOL-INTRARE
    .
    .
    .
```

12.28 *Î.* Care din variantele de mai jos este mai simplă?
a) MOVE ART—CART TO ART—BAND
WRITE ART—BAND
b) WRITE ART—BAND FROM ART—CART
R. b

12.29 Zona definită prin *nume—dată* poate fi descrisă în WORKING—STORAGE SECTION sau în FILE SECTION.

12.30 Pentru crearea fișierelor cu organizare secvențială—indexată sau selectivă și pentru actualizarea lor se utilizează formatul:

WRITE nume—articol [**FROM** nume—dată]
INVALID KEY instrucțiune—imperativă ....

12.31 La crearea fișierelor cu organizare secvențială—indexată articolele trebuie furnizate în ordinea crescătoare a valorilor luate de cîmpul cheie. Cînd această ordine este respectată are loc operația de scriere a articolului în fișier; în caz contrar articolul nu se scrie pe suport și controlul este trecut instrucțiunii (instrucțiunilor) precizate prin clauza INVALID KEY.

Exemplu:

```
WRITE ART-DISC INVALID KEY
              DISPLAY 'EROARE SECVENTA' COD
              MOVE 1 TO V-IVK.
```

unde:
- V—IVK, un cîmp care ia valoarea 1 cînd se detectează cazul „cheie invalidă".

12.32 *Î.* Să presupunem că într-un fișier pe bandă magnetică articolele se află ordonate după valorile cîmpului cheie astfel: 15750, 15752, 15801, 15754, 15805, 15804, 1830. Care articole, la executarea instrucțiunii WRITE, nu se înregistrează în fișierul pe disc magnetic organizat secvențial—indexat?

. . . .

R. 15801
15804

12.33 Crearea fișierelor cu organizare selectivă precum și actualizarea fișierelor cu organizare secvențială—indexată și selectivă se face în acces direct.

12.34 *Instrucțiunea WRITE*, pentru crearea fișierelor cu organizare selectivă determină scrierea articolului în caseta al cărei număr de ordine corespunde

cu valoarea datei precizate prin ACTUAL KEY ; are loc scrierea articolului în fişier :

— în partea principală a fişierului dacă există loc suficient ;

— în partea de depăşire, dacă în partea principală nu mai este loc.

După scrierea articolului, controlul este preluat de către instrucţiunile din fraza următoare.

Articolul nu se scrie în fişier atunci cînd :

— în fişier (partea principală sau de depăşire) mai există încă un articol cu aceeaşi cheie ;

— în fişierul respectiv nu există o casetă a cărei adresă este egală cu valoarea cheii reale. Controlul este preluat în acest caz de către instrucţiunile precizate prin clauza INVALID KEY.

12.35 Actualizarea fişierelor cu organizare selectivă prin instrucţiunea WRITE constă în adăugarea de articole şi este asemănătoare cu operaţiunea de creare

Actualizarea fişierelor cu organizare secvenţială—indexată, prin adăugare de articole, se realizează de asemenea cu ajutorul instrucţiuni WRITE.

12.36 *I.* Instrucţiunea WRITE utilizată pentru a scrie într-un fişier cu organizare secvenţială—indexată cuprinde clauza . . . . care indică instrucţiunea ce trebuie executată în cazul în care scrierea nu poate avea loc. La crearea acestor fişiere articolele trebuie furnizate în ordinea . . . . a valorii cheii articolului. Fişierele cu organizare secvenţială—indexată se crează în acces . . . . Crearea fişierelor organizate selectiv precum şi actualizarea fişierelor organizate secvenţial—indexat şi selectiv se face în acces . . . .

R. — INVALID KEY
— crescătoare
— secvenţial
— direct

12.37 *I.* Instrucţiunile READ şi WRITE pentru fişiere cu organizare secvenţială—indexată şi selectivă sînt instrucţiuni condiţionale pentru faptul că au clauza . . . .

R. INVALID KEY

12.38 *I.* Opţiunea FROM din cadrul instrucţiunii WRITE înlocuieşte instrucţiunea . . . .

R. MOVE

12.39 *Instrucţiunea REWRITE*, permite realizarea operaţiilor de rescriere într-un fişier a articolelor la care s-au făcut modificări de conţinut.

Formatul simplificat este :

REWRITE nume—articol
[**FROM** — nume—dată]
[**INVALID** KEY instrucţiune ....]

Utilizarea instrucţiunii REWRITE presupune realizarea următoarelor operaţii :

— citirea articolului de actualizat ;

— realizarea modificărilor privind conţinutul anumitor date din structura articolului;

— rescrierea articolului actualizat.

Exemplu:

```
MOVE VALOARE—CHEIE TO CHEIE—SIMBOLICA
READ FISIER—SI INVALID KEY MOVE 1 TO IVK.
IF IVK = 0 MOVE . . . . . . . . . . . . . . . . . .
          . . . . . . . . . . . . . . . . . . . . . .
          REWRITE ARTICOL  . . . . . . . . .
          ELSE MOVE O TO IVK.
```

12.40 Instrucţiunile precizate prin clauza INVALID KEY din instrucţiunea REWRITE au rolul de a controla dacă nu s-a operat o modificare asupra valorii cheii de adresare.

Instrucţiunea DELETE, permite realizarea operaţiilor de ştergere a articolelor dintr-un fişier. Formatul simplificat este:

DELETE nume—articol

Operaţia de ştergere se realizează prin invalidare, în scopul de a nu distruge înlănţuirea articolelor (pentru organizarea secvenţială—indexată). Anularea unui articol (prin invalidare) presupune mai întîi citirea lui.

Exemplu:

```
MOVE VALOARE—CHEIE TO CHEIE—SIMBOLICA
READ FISIER—SI INVALID KEY MOVE 1 TO IVK.
IF IVK = 0
    THEN
        DELETE ARTICOL
    ELSE
        MOVE 0 TO IVK
        DISPLAY 'ART DE ANULAT INEXISTENT'.
```

Pentru verificarea cunoştinţelor rezolvaţi testele 5 şi 6, paginile 358, 361.

13.5 Într-o primă etapă problema supusă studiului se descompune în module obţinîndu-se aşa cum am arătat în fig. 13.1, o organigramă pe module şi un număr de specificaţii asociate acestora (*specificaţii de intrare, specificaţii de prelucrare* şi *specificaţii de ieşire*). Organigrama pe module reprezintă componentele programului, poziţiile lor ierarhice şi relaţiile între acestea.

13.6 Orice apel între module trebuie să se efectueze de sus în jos şi în funcţie de structura stabilită. Aşa cum s-a prezentat în fig. 13.1 modulul A nu poate apela direct decît modulele B, C şi D.

13.7 În procesul de modularizare se disting două mari categorii de module şi anume:

— *module de comandă* (*directoare*), care realizează în principiu operaţii de iniţializare a datelor intermediare, de control şi de transfer a controlului către modulele de prelucrare, aceste module vor conţine cu prioritate instrucţiuni IF şi PERFORM (sau CALL);

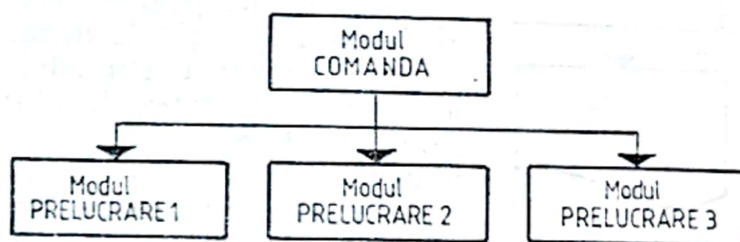


Fig. 13.2.

— *module de prelucrare*, care realizează funcţiile de bază ale programului; un modul de prelucrare este activat (apelat) de un modul ierarhic superior iar după realizarea funcţiei redă controlul modulului apelant (fig. 13.2).

13.8 𝒥. Problemele complexe pentru a fi programabile trebuie mai întîi descompuse în a) . . . . aplicînd în practică criteriile b) . . . .

ℛ. a) module

b) — omogenitatea funcţiilor

— utilizarea diverselor fişiere

— păstrarea unor dimensiuni acceptabile.

13.9 𝒥. O funcţie cuprinde . . . . executabile în condiţii . . . . şi se poate exprima matematic . . . .

ℛ. — un grup de operaţii

— similare

— $y = f_p(x)$

13.10 Fie „x" numele problemei (supusă studiului în vederea programării) care constă în prelucrarea datelor privind intrările de valori materiale în vederea obţinerii informaţiilor la nivel operaţie de intrare, fel de material, gestiune şi întreprindere (fig. 13.3).

13.11 Structura situaţiei la imprimantă este prezentată în fig. 13.4.

---

a) A apela un modul înseamnă a trece controlul instrucţiunilor ce-l definesc şi a reveni după execuţia acestora la modulul apelant (la instrucţiunea imediat următoare).

13.12 Din fig. 13.4 rezultă că în cadrul situaţiei intrărilor de materiale informaţiile se pot grupa:

— informaţii pentru a caracteriza intrările de materiale la nivel întreprindere:

TOTAL—VALOARE—INTREPRINDERE

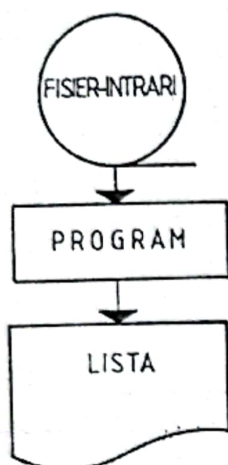


Fig. 13.3.

— informaţii pentru a caracteriza intrările de materiale la nivel gestiune:

TOTAL—VALOARE—GESTIUNE

— informaţii pentru a caracteriza intrările de materiale la nivel fel material:

TOTAL—CANTITATE

TOTAL—VALOARE—FEL—MATERIAL

— informaţii pentru a caracteriza intrările de materiale la nivel operaţie de intrare.

13.13 Prin corelarea studiului structurii informaţiei în ieşire cu modalitatea de obţinere se ajunge la schema din fig. 13.5.

unde:

n, numărul gestiunilor dintr-o întreprindere;

m, numărul felurilor de materiale dintr-o gestiune;

k, numărul articolelor care se referă la acelaşi fel de material (el desemnează şi numărul articolelor care formează o grupă de control).

SITUATIA INTRARILOR DE MATERIALE

NR CRT	GESTIUNEA	FELUL MATERIALULUI	DENUMIREA MATERIALULUI	UM	CANTITATEA	PRETUL UNITAR	VALOAREA
1	02	2356	BETON B75	MC	10.00	235.00	2350.00
2	02	2356	BETON B75	MC	20.00	235.00	4700.00
		TOTAL FEL MATERIAL			30.00		7050.00
3	02	2358	BETON 250	MC	20.00	265.00	5300.00
4	02	2358	BETON 250	MC	30.00	265.00	7950.00
		TOTAL FEL MATERIAL			50.00		13250.00
...	...	...	...	...	...	...	...
		TOTAL GESTIUNE					43500.00
79	03	3456	BITUM B14	TO	5.00	720.00	3600.00
80	03	3456	BITUM B14	TO	10.00	720.00	7200.00
81	03	3456	BITUM B14	TO	20.00	720.00	14400.00
		TOTAL FEL MATERIAL			35.00		25200.00
...	...	...	...	...	...	...	...
		TOTAL FEL MATERIAL			40.00		24000.00
		TOTAL GESTIUNE					1345000.00
		TOTAL INTREPRINDERE					15675025.00

Fig. 13.4.

13.14 Pe structura din fig. 13.5 se va axa și procesul de descompunere în module prezentat în fig. 13.6, 13.7, 13.8, 13.9.

Modulul MODUL—COMANDA inițializează datele care permit obținerea totalurilor la nivel întreprindere, activează conform unei structuri repetitive modulul GESTIUNE, iar cînd nu mai sînt date în fișier, afi-

TOTAL-CANTITATE ← TOTAL-CANTITATE + CANTITATE
TOTAL-VALOARE-FEL-MATERIAL ← TOTAL-VALOARE-FEL-MATERIAL + + VALOARE
TOTAL-VALOARE-GESTIUNE ← TOTAL-VALOARE-GESTIUNE + + TOTAL-VALOARE-FEL-MATERIAL
TOTAL-VALOARE-INTREPRINDERE ← TOTAL-VALOARE-INTREPRINDERE + + TOTAL-VALOARE-GESTIUNE
NUMAR-CURENT ← NUMAR-CURENT + 1
VALOARE ← CANTITATE * PRET-UNITAR

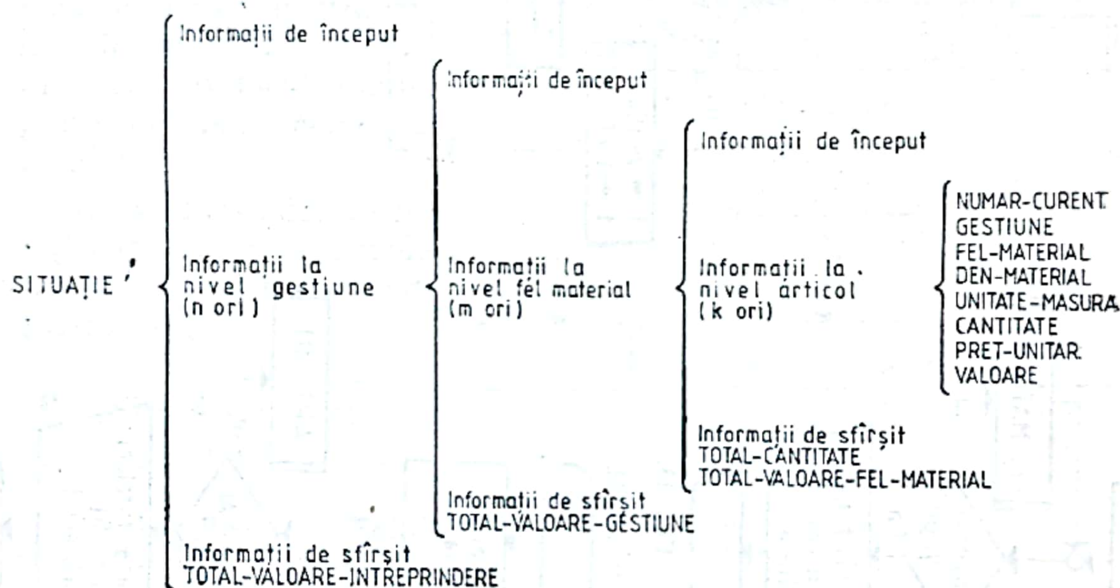


Fig. 13.5.

șează totalurile. La rîndul lui modulul GESTIUNE inițializează datele care permit obținerea totalurilor la nivel gestiune și activează conform unei structuri repetitive modulul FEL—MATERIAL, iar cînd s-au terminat articolele care se referă la aceeași gestiune afișează informațiile descrise prin specificațiile problemei. În mod asemănător modulul FEL—MATERIAL realizează prelucrările pentru nivelul ierarhic al grupei de articole care au aceeași valoare pentru cîmpul FEL—MATERIAL.

Modulul ARTICOL cuprinde operații ce se pot grupa în:

a) operații de calcul;

b) operații de control pentru a determina revenirea la modul apelant.

13.15 În figurile 13.10a și 13.10b sînt prezentate alte variante de structurare pe module a problemei din fig. 13.4.

Cu toată rigurozitatea introdusă prin utilizarea structurilor standard de control (pe diverse nivele ierarhice ale conceperii programelor) nu se

MODUL-COMANDA

Început

Inițializări Variabile

Aducerea fișierelor în stare operațională

Citește FIȘIER-INTRARI

EOF ?

Da

Nu

1 → V-EOF

GESTIUNE

V-EOF=1 ?

Nu

Da

Afișează totaluri la nivel întreprindere

Pregătirea fișierelor pentru încheierea procesului de prelucrare

Sfîrșit

Fig. 13.6

GESTIUNE

Inițializări variabile

FEL-MATERIAL

V-GEST=1 ?

Nu

Da

Afișează totaluri la nivel gestiune

Fig. 13.7

FEL-MATERIAL

Inițializări variabile

ARTICOL

V-FEL-MAT =1 ?

Nu

Da

Afișează totaluri la nivel fel material

← Fig. 13.9

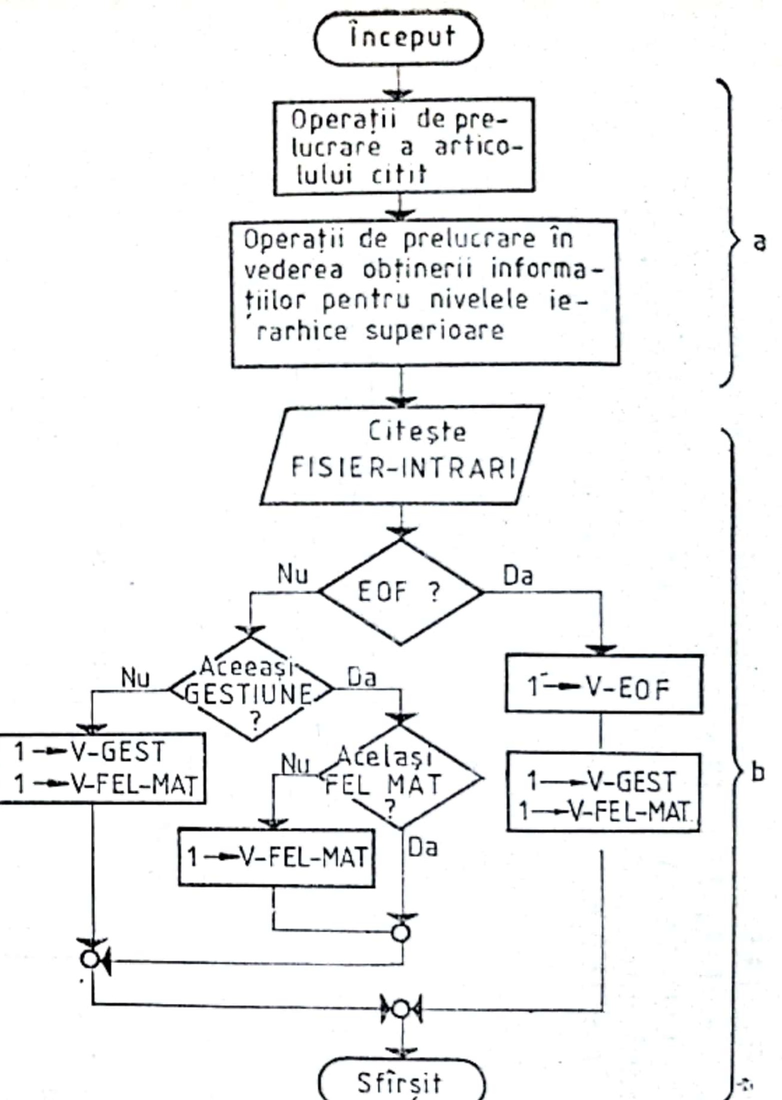


Fig. 13.8

poate vorbi încă de existența unei relații funcționale, prin care unei probleme date să-i asocieze un singur program și numai unul.

13.16 *Instrucțiunea PERFORM* permite așa cum am văzut în 6.8 descrierea unui ciclu. La întîlnirea acestei instrucțiuni se transferă controlul la prima instrucțiune din procedura apelată iar la sfîrșitul executării de una, sau dacă activarea este repetitivă, de mai multe ori se revine la instrucțiunea imediat următoare. Folosirea instrucțiunii PERFORM impune gruparea instrucțiunilor în paragrafe și eventual secțiuni identificabile printr-un *nume—procedură* unic.

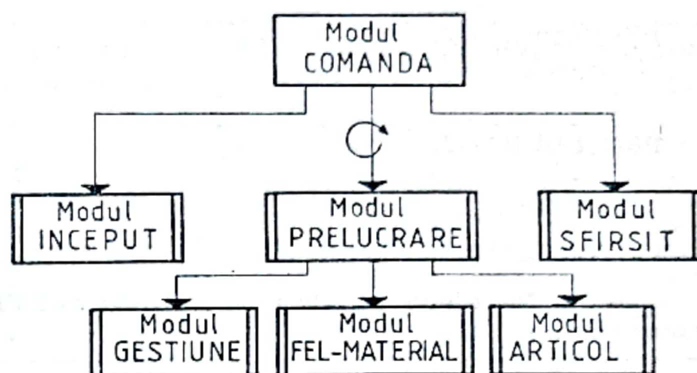


Fig. 13.10. a

13.17 În cazul apelului necondiționat structura simplificată a instrucțiunii PERFORM este:

PERFORM nume—procedură

La întîlnirea instrucțiunii PERFORM se realizează:

— saltul la paragraful *nume—procedură*;

— execuția secvenței de instrucțiuni din cadrul procedurii;

— revenirea la instrucțiunea imediat următoare instrucțiunii PERFORM.

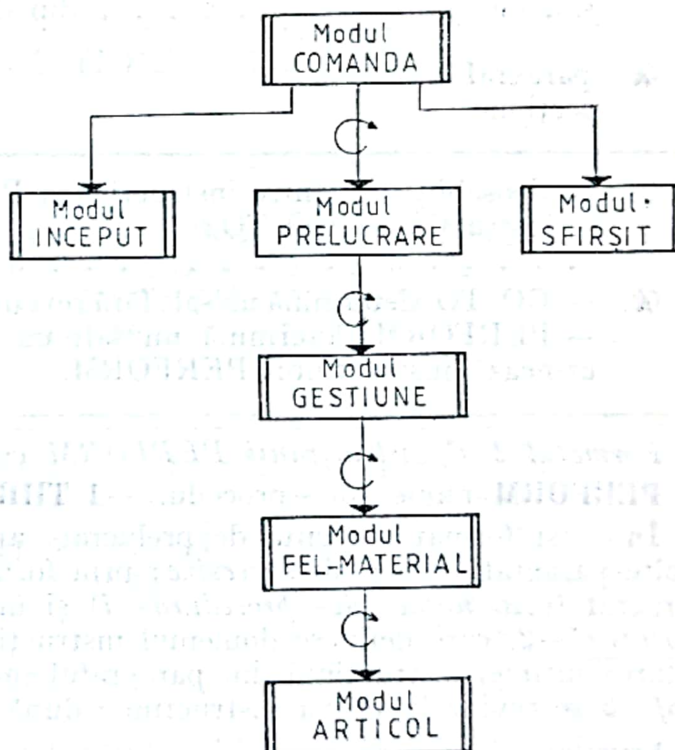


Fig. 13.10. b

Exemplu:

```
    PERFORM INITIALIZĂRI
IF COD—CARTELA = 1
    PERFORM—PROCEDURA—1
ELSE IF COD—CARTELA = 2
    PERFORM PROCEDURA—2
ELSE IF COD—CARTELA = 3
    PERFORM PROCEDURA—2
```

```
        ELSE PERFORM PROCEDURA—EROARE
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
PROCEDURA—1.
    .
    .
    .
PROCEDURA—2.
    .
    .
    .
PROCEDURA—3.
    .
    .
    .
INITIALIZĂRI.
```

Observație Procedurile apelate prin instrucțiunea PERFORM nu trebuie, în general, să urmeze fizic acesteia.

13.18 **Î.** Nume—de—procedură este un nume de . . . . sau de . . . . din diviziunea de procedură.

R. paragraf
secțiune

13.19 **Î.** Ce deosebire este între instrucțiunea PERFORM în formatul 13.17 și instrucțiunea GO TO?

. . . .

R. — GO TO determină un salt fără revenire la instrucțiunea următoare
— PERFORM determină un salt cu revenire la instrucțiunea ce urmează instrucțiunii PERFORM.

13.20 *Formatul 2 al instrucțiunii PERFORM* este:

PERFORM nume—de—procedură—1 **THRU** nume—de—procedură—2

În acest format secvența de prelucrare apelată este formată din mai multe paragrafe (secțiuni) *succesive*; prin formatul 2 se precizează primul paragraf (prin *nume—de—procedură—1*) și ultimul paragraf (*nume—de—procedură—2*) care definesc domeniul instrucțiunii PERFORM. După executarea ultimei instrucțiuni din paragraful precizat prin *nume—de—paragraf—2* se revine la prima instrucțiune după PERFORM.

Exemplu:

```
        . . . . . . . . . . . . . . . . . . .
        PERFORM    P2    THRU    P3
        ADD    1    TO    NRC.
P1. . . . . . . . . . . . . . . . . . . .
        . . . . . . . . . . . . . . . . . . .
P2. . . . . . . . . . . . . . . . . . . .
        . . . . . . . . . . . . . . . . . . . . . . . . . . .
P3. . . . . . . . . . . . . . . . . . . .
```

Observație Folosirea opțiunii THRU se impune îndeosebi cînd prin testarea unei condiții în cadrul unei proceduri se decide abandonarea instrucțiunilor ce urmează celei de testare.

13.21 *Î.* Precizați instrucțiunea PERFORM pentru executarea procedurilor succesive PROC—1, PROC—2, PROC—3, PROC—4, PROC—5 și PROC—6

. . . .

R. PERFORM PROC—1 THRU PROC—6

13.22 *Î.* Dacă nu se utilizează formatul cu THRU cum se mai poate realiza o secvență program cu același efect?

. . . .

R.
```
PERFORM PROC—1
PERFORM PROC—2
PERFORM PROC—3
PERFORM PROC—4
PERFORM PROC—5
PERFORM PROC—6
```

12.23 *Î.* Scrieți secvența care traduce în instrucțiuni COBOL operațiile prevăzute în schema din figura 13.11 folosind opțiunea THRU

. . . .

R.
```
PERFORM I—MODUL THRU S—MODUL
.
.
.
I—MODUL.
        MULTIPLY CANT BY PRET GIVING VALOARE
        IF LUNA NOT  = 10 GO TO S—MODUL.
        ADD VALOARE TO TOTAL—VALOARE
        DISPLAY 'VALOARE = ' VALOARE.
S—MODUL.
        EXIT.
```

13.24 *Formatul 3 al instrucțiunii PERFORM* este:

PERFORM nume—de—procedură—1
  [**THRU** nume—de—procedură—2]

  { literal numeric / nume—dată } **TIMES**

Conform formatului 3 se poate cere execuția procedurii *nume—de—procedură—1* (după caz a procedurilor cuprinse între *nume—de—procedură—2* și *nume—de—procedură—2*) de un număr de ori egal cu *literal—numeric* sau conținutul cîmpului *nume—dată*, după care se revine la prima instrucțiune următoare din program (de după PERFORM). Prin urmare formatul 3 al instrucțiunii PERFORM permite

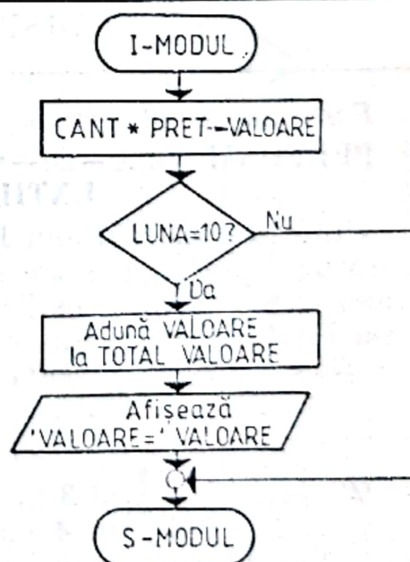


Fig. 13.11.

realizarea ciclului cu număr fix de iterații și furnizează informații cu privire la :
— corpul ciclului ;
— modalitatea de a controla iterațiile.

Corpul unui ciclu în limbajul COBOL reprezintă una sau mai multe instrucțiuni indicată (e) în instrucțiunea PERFORM prin *nume—procedură* sau *nume—procedură—1* THRU *nume—procedură—2*

```
                PERFORM REDARE 4 TIMES
                MOVE A TO B
                . . . . . . . . . . . . . . . .
                . . . . . . . . . . . . . . . .
REDARE.
                WRITE ART1 FROM ZONA AFTER 1.
```

Instrucțiunea PERFORM trece controlul procedurii REDARE; după execuția de 4 ori a instrucțiunilor care compun procedura REDARE se revine la instrucțiunea MOVE (prima de după PERFORM).

13.25 Literalul numeric sau conținutul cîmpului definit prin *nume—de—dată* trebuie să fie număr întreg > 0.

Î. Să presupunem că într-un program este nevoie ca la anumite
13.26 intervale să se editeze un număr de rînduri libere la imprimantă (exemplu 4). În acest scop se utilizează instrucțiunea DISPLAY. Vă rugăm să scrieți instrucțiunea PERFORM și paragraful care realizează acest spațiu liber

. . . .
. . . .
. . . .

R.
```
PERFORM INTERVAL 4 TIMES
. . . . . . . . . . . . . . . . . . . 
INTERVAL.
            DISPLAY SPACES.
```

13.27 *Formatul 4 al instrucțiunii PERFORM* este :

PERFORM nume—de—procedură—1 [**THRU** nume—de—procedură—2]
UNTIL condiție

Utilizarea instrucțiunii PERFORM în acest format are ca efect executarea unei proceduri (care poate fi formată din unul sau mai multe paragrafe) atîta timp cît condiția este falsă. În formatul cu UNTIL instrucțiunea PERFORM permite realizarea ciclurilor cu număr variabil de iterații.

13.28 *Î.* Care este diferența între formatul 3 și formatul 4 ?

. . . .
. . . .

R. — în formatul 3 numărul de repetări este dat de la început ;
— în formatul 4 numărul de repetări este determinat prin verificarea unei condiții.

13.29 Se consideră, următorul exemplu :

```
. . . . . . . . . . . . . . . . . .
      MOVE 4 TO ZONA—1
      MOVE 7 TO ZONA—2
      PERFORM P1 UNTIL ZONA—1 = ZONA—2
. . . . . . . . . . . . . . . . . .
P1. MULTIPLY ZONA—1 BY ZONA—2 GIVING ZONA—3
    ADD 1 TO ZONA—1.
```

Secvenţa de prelucrare căreia i se trece controlul (paragraful P1) va fi executată pînă cînd ZONA—1 = ZONA—2

ZONA—1	ZONA—2	ZONA—1 = ZONA—2	P1
4	7	NU	DA
5	7	NU	DA
6	7	NU	DA
7	7	DA	NU

Paragraful P1 va fi executat de trei ori.

13.30 *Formatul 5 simplificat al instrucţiunii PERFORM* este :

PERFORM nume—de—procedură—1 [**THRU** nume—de—procedură—2]

VARYING nume—cîmp—1 **FROM** {nume—cîmp—2 | literal—numeric—1} **BY**

{nume—cîmp—3 | literal—numeric—2} **UNTIL** expresie—condiţională

Instrucţiunea realizează executarea condiţionată a unei proceduri de prelucrare ce constă dintr-unul sau mai multe paragrafe (secţiuni) ; *nume—cîmp—1* reprezintă un contor care se iniţializează cu valoarea dată de *nume—cîmp—2* sau *literal—numeric—1*. Raţia de creştere este dată prin *nume—cîmp—3* sau *literal—numeric—2*. Dacă la realizarea testării valorii expresiei condiţionale răspunsul este negativ se execută procedura indicată iar în caz afirmativ se revine la instrucţiunea care urmează după PERFORM (fig. 13.12).

Exemplu:

```
PERFORM CALCUL VARYING I FROM 1
        BY 1 UNTIL I > 12;
```

unde:

contorul I ia valori de la 1 (valoarea iniţială) cu pasul 1 pînă la 12 (inclusiv). Se va executa procedura CALCUL pînă cînd se îndeplineşte condiţia I > 12 (vezi şi lecţia 15).

Instrucţiunea PERFORM în formatul 5 se utilizează, în general, cînd datele sînt descrise cu ajutorul clauzei OCCURS. Din acest motiv asupra ei vom reveni în lecţia 15.

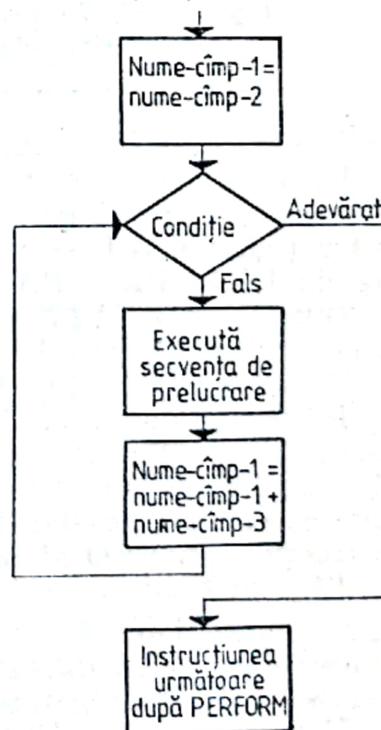


Fig. 13.12.

Observație:
— procedura (paragraful, secțiunea) apelată printr-o instrucțiune PERFORM nu trebuie (în principiu) să urmeze acesteia;
— folosirea formatului cu opțiunea THRU cere ca paragrafele (secvențele) apelate să fie grupate deoarece logica derulării programului este dependentă de structura fizică a acestora;
— în cazul în care o procedură (secvență sau paragraf) apelată prin PERFORM conține o altă instrucțiune PERFORM, aceasta trebuie fie să fie inclusă în totalitate în procedură fie exterioară acesteia;
— o procedură apelată prin instrucțiunea PERFORM nu poate conține instrucțiunea PERFORM care face apel la ea însăși;
— o procedură apelată prin instrucțiunea PERFORM poate include parțial o altă procedură apelată la rîndul său de o altă instrucțiune PERFORM cu precizarea că nici una din cele două proceduri nu trebuie să conțină o instrucțiune PERFORM care o cheamă pe cealaltă.
— o procedură apelată de o anumită instrucțiune PERFORM poate fi apelată și de alte instrucțiuni PERFORM;
— o procedură apelată de o instrucțiune PERFORM poate fi executată și prin derularea secvențială a programului.

13.31 *Î*. O secvență de prelucrare apelată de o instrucțiune PERFORM poate conține instrucțiunea PERFORM?

. . . .

R. DA

13.32 *Î*. Se apelează cu instrucțiunea PERFORM o secvență de prelucrare. Această secvență poate să conțină o instrucțiune PERFORM care face apel la ea însăși?

. . . .

R. NU

13.33 *Î*. O secvență de prelucrare apelată prin instrucțiunea PERFORM, mai poate fi apelată și de alte instrucțiuni PERFORM

. . . .

R. DA

13.34 Definirea unui punct comun de ieșire pentru mai multe secvențe de prelucrare sau precizarea sfîrșitului logic al unui subprogram * apelat prin instrucțiunea CALL se realizează *prin instrucțiunea EXIT* care are formatul: EXIT PROGRAM.

13.35 *Instrucțiunea EXIT* se utilizează cu frecvența mare în legătură cu instrucțiunile PERFORM și CALL sau cu o procedură declarată cu fraza USE (vezi și lecția 17) **.

13.36 Încheierea unei proceduri apelate prin PERFORM cu un paragraf care conține numai instrucțiunea EXIT se recomandă atunci cînd este necesar să se întrerupă suita de instrucțiuni apelată prin PERFORM înainte de terminarea ei normală. În secvența apelată prin PERFORM printr-o instrucțiune GO TO se realizează saltul la punctul de ieșire definit prin EXIT.

---

* Un subprogram traduce în construcții COBOL o parte din algoritmul de rezolvare a unei lucrări complexe; el este compilat independent și asociat programului în faza de editare.

** Fiind instrucțiune vidă ea poate fi utilizată oriunde în program (de exemplu pentru artificii stilistice).



Exemplu:

```
. . . . . . . . . . . . . . . . . . . . . .
      PERFORM P1 THRU P3
. . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . .
P1. . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . .
    IF COD-MAT = COD - MATM GO TO P3.
. . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . .
P3. EXIT.
P4. . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . .
```

13.37 Instrucţiunea EXIT atunci cînd este întîlnită fără să fie sub controlul unei instrucţiuni PERFORM devine inefectivă.

13.38 EXIT este singura instrucţiune a paragrafului în care se utilizează. Instrucţiunea EXIT este un element stilistic care defineşte un punct unic de ieşire dintr-un modul; ultimul paragraf dintr-un PERFORM—THRU etc. Prin EXIT se pot însera paragrafele fictive de ieşire la care se poate realiza saltul cu ajutorul instrucţiunii GO TO fără a executa întreaga procedură.

13.39 Formatul EXIT cu opţiunea PROGRAM se utilizează în legătură cu instrucţiunea CALL (vezi lecţia 17).

13 40 *Î.* Se dă următoarea secvenţă de prelucrare (fig. 13.13):
Vă rugăm să scrieţi instrucţiunea prin care se trece controlul acestei secvenţe cu revenirea (la sfîrşitul executării ei) la instrucţiunea imediat următoare din program.

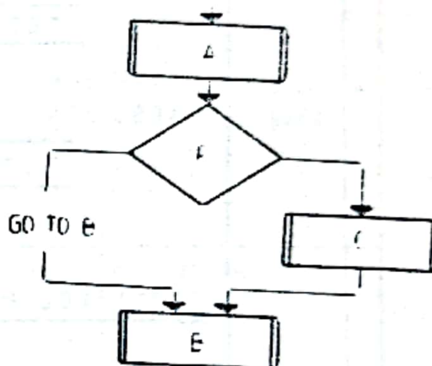


Fig. 13.13.

. . . .

R. PERFORM A THRU B

13.41 *Î.* Descompuneţi în module şi respectiv operaţii elementare procesul de prelucrare automată a datelor în vederea obţinerii situaţiei cîştigurilor în acord pentru fiecare om al muncii cu grupare valorică pe secţii iar în cadrul acestora pe locuri de muncă (fig. 13.14).

Pentru rezolvare se dau următoarele explicaţii suplimentare:

a) la proiectarea programului veţi avea în vedere structura pe nivele ierarhice prezentate în fig. 13.15 şi structura datelor din figura 13.16;

b) articolele memorate în FIŞIER—LUCRU—ACORD sînt supuse prelucrării pe grupe numite şi grupe de control; o grupă de control include articolele care se referă la aceeaşi valoare a unui indicator de grupare (exemple: grupă de control la nivel secţii, grupa de control la nivel loc muncă şi grupa de control la nivel număr marcă);

c) la realizarea calculelor în vederea obținerii informațiilor privind cîștigurile din situația prezentată în fig. 13.14 se utilizează cîmpurile:

• T—CIȘTIG—ACORD—MARCA, care va memora suma cîștigurilor în acord la nivel număr marcă (om al muncii);

• T—CÎȘTIG—LOC—MUNCA, care va memora suma cîștigurilor în acord la nivel loc de muncă;

SITUATIA CISTIGURILOR IN ACORD

NR CRT	NUMAR MARCA	NUME SI PRENUME	DOCUMENT FEL M	DOCUMENT DATA	CANTITATE EXECUTATA	TARIF PE BUCATA	CISTIG IN ACORD
1	4335	ALBU NICOLAE	5134	05.01	13.00	20.00	260.00
2	4335	ALBU NICOLAE	5140	10.01	20.00	30.00	600.00
3	4335	ALBU NICOLAE	5160	29.01	60.00	40.00	2400.00
		TOTAL					3260.00
4	4340	BARBU ION	5156	06.01	30.00	20.00	600.00
5	4340	BARBU ION	5150	29.01	90.00	25.00	2250.00
		TOTAL					2850.00
		TOTAL LOC MUNCA					25843.00
		TOTAL SECTIE					184535.00
		TOTAL INTREPRINDERE					

Fig. 13.14.

• T—CÎȘTIG—SECȚIE, care va memora suma cîștigurilor în acord la nivel secție;

• T—CÎȘTIG—ÎNTREPRINDERE, care va memora suma cîștigurilor în acord la nivel întreprindere;

• V—EOF, care semnalează detectarea mărcii de sfîrșit de fișier;

• V—SECŢIE, care semnalează prin valoarea 1 trecerea la o nouă valoare a variabilei SECŢIE din articolul asociat fişierului în intrare;

• V—LOC—MUNCA, care semnalează prin valoarea 1 trecerea la o nouă valoare a variabilei LOC—MUNCA din articolul asociat fişierului în intrare;

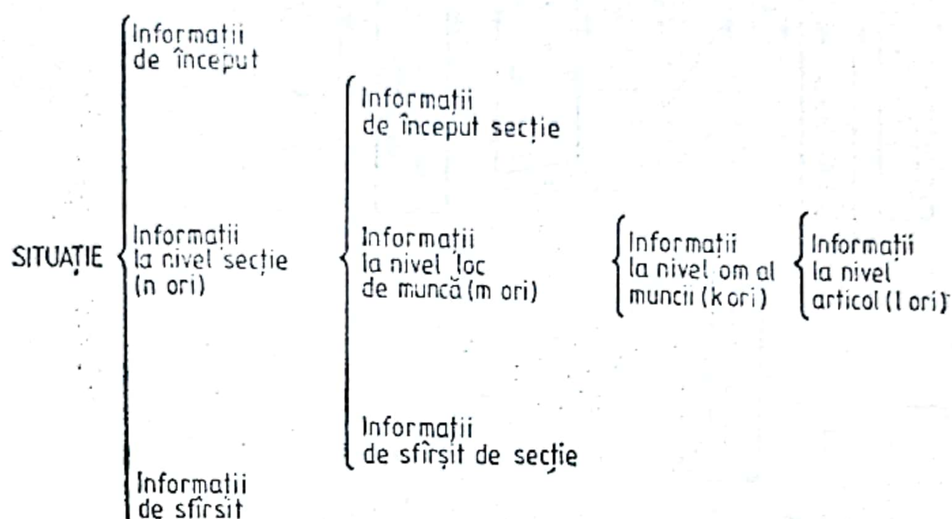


Fig. 13.15.



ARTICOL

SECTIE	LOC MUNCA	NUMAR MARCA	NUME SI PRENUME	DOCUMENT		CANTITATE EXECUTATA	TARIF PE BUCATA
				NR	DATA		
n	n	n	a-n	n	n	n	n
2	4	4	40	5	4	5	2

Fig. 13.16.

• V—NUMĂR—MARCA, care semnalează prin valoarea 1 trecerea la o nouă valoare a variabilei NUMĂR—MARCA din articolul asociat fişierului de intrare. Corectaţi schema din fig. 13.30 prin înlocuirea variabilei V—MARCA cu variabila V—NUMAR—MARCA.

R. (fig. 13.17, 13.18, 13.19, 13.20, 13.21)

Răspunsurile sînt aproximativ asemănătoare?

DA → 13.42
NU ↰ 5

13.42 Pentru a mări performanţele sistemului prin reducerea redundanţelor vă propunem utilizarea în intrare a două fişiere (fig. 13.22):

— *FISIER—PERSONAL*, care să conţină datele permanente referitoare la fiecare om al muncii din întreprindere şi care are organizare secvenţială—indexată;

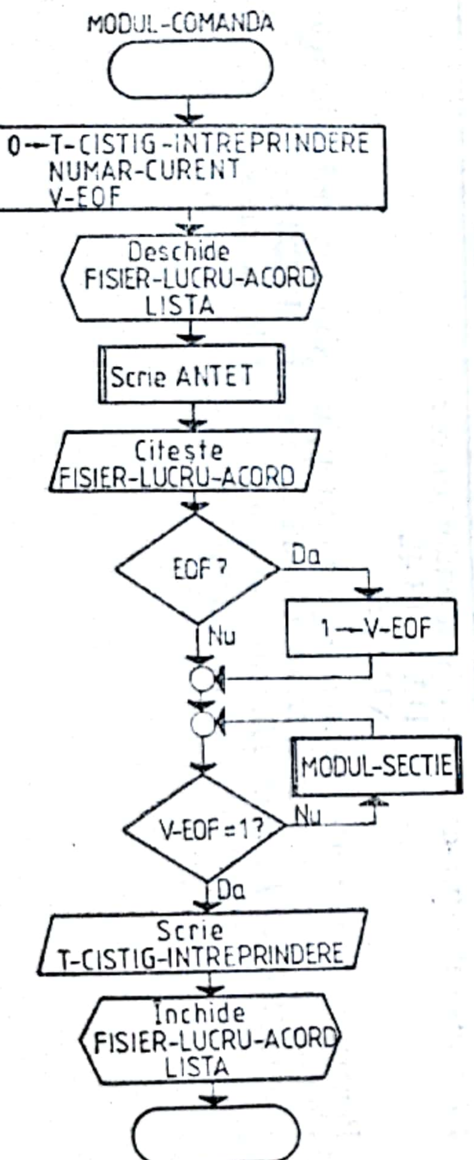


Fig. 13.17.

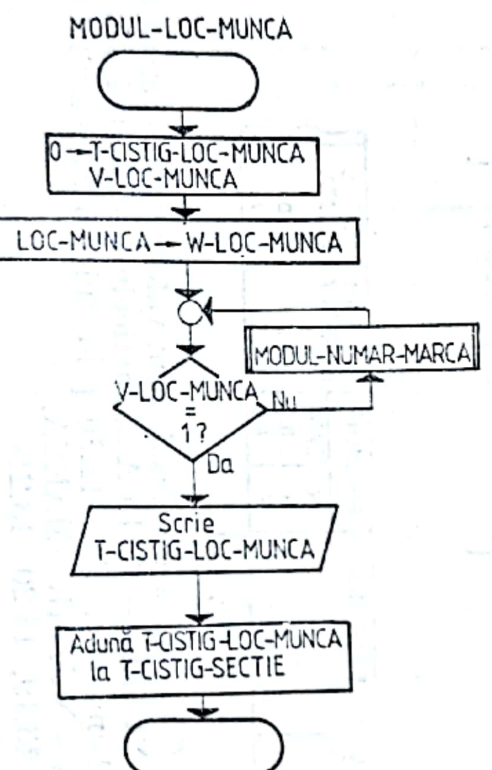


Fig. 13.18.

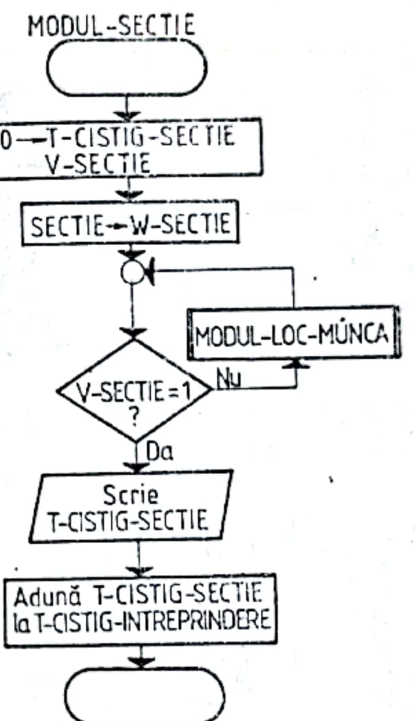


Fig. 13.19.

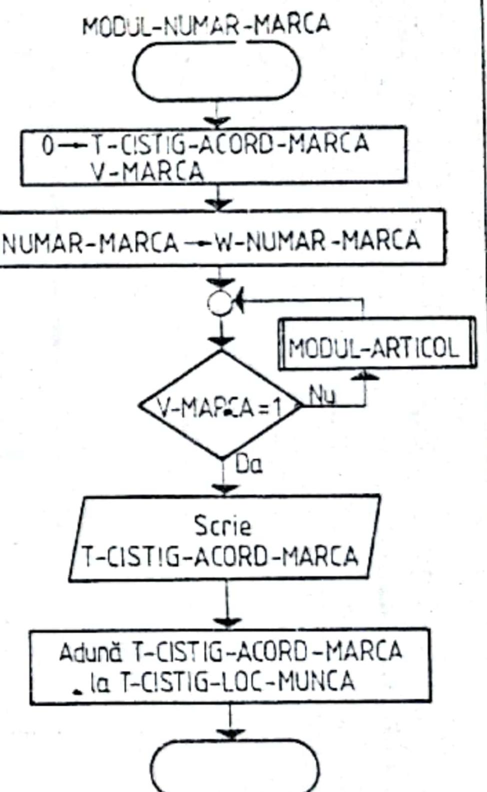


Fig. 13.20.

MODUL-ARTICOL

CISTIG-ACORD ← CANTITATE-EXECUTATA * TARIF-PE-BUCATA

T-CISTIG-ACORD-MARCA ← T-CISTIG-ACORD-MARCA + CISTIG-ACORD

NUMAR-CURENT ← NUMAR-CURENT + 1

Pregăteşte imagine articol de imprimat în LISTA

Scrie articol în LISTA

Citeste FISIER-LUCRU-ACORD

EOF ?

Nu

Da

SECTIE = W-SECTIE ?

1 → V-EOF

1 → V-SECTIE

1 → V-LOC-MUNCA

1 → V-NUMAR-MARCA

1 → W-SECTIE
V-LOC-MUNCA
V-NUMAR-MARCA

LOC-MUNCA = W-LOC-MUNCA ?

1 → V-LOC-MUNCA
V-NUMAR-MARCA

NUMAR-MARCA = W-NUMAR-MARCA ?

1 → NUMAR-MARCA

Fig. 13.21.

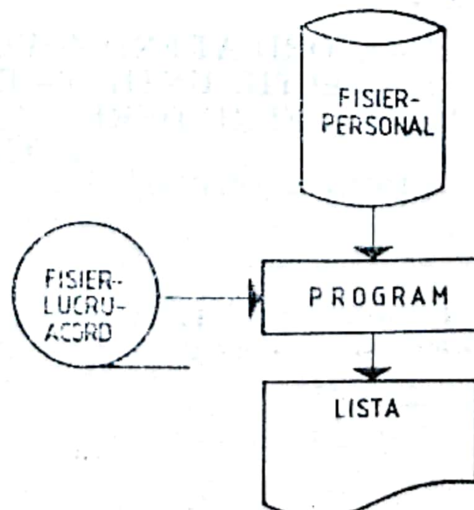


Fig. 13.22.

— *FISIER—LUCRU—ACORD* care să conțină datele variabile (tranzacțiile) referitoare la cîștigurile în acord.

13.43 În structurarea procesului de prelucrare a datelor fișierului FIȘIER—LUCRU—ACORD joacă rol principal, spunem că el constituie *fișierul pilot* al programului de proiectat (conținutul lui indică ce date din celelalte fișiere trebuie citite în vederea prelucrării).

13.44 În situația obținută la imprimantă ne propunem ca numele și prenumele pentru fiecare om al muncii să apară o singură dată; în continuare pentru un om al muncii să fie imprimate atîtea rînduri cîte articole din fișierul FIȘIER—LUCRU—ACORD au aceeași valoare pentru cîmpul MARCA.

13.45 *Î* Descrieți numai modulele care au suferit modificări în condițiile folosirii fluxului general al datelor din fig. 13.22 și a machetelor asociate fișierelor în intrare (fig. 13.23; 13.24). Care sînt erorile din 13.25?

R. (fig. 13.25; 13.26).
Din figura 13.25. rezultă că a fost necesară includerea în modulul MODUL—NUMĂR—MARCĂ a operațiilor care vizează accesul la datele permanente și controlul privind existența acestora. În ceea ce privește modulul MODUL—ARTICOL el cuprinde în primul rînd testul V—IVK — 1 pentru a vedea dacă prelucrarea datelor din articolul citit are sens. Cum este și firesc nu vor fi supuse prelucrării decît acele date din FISIER—LUCRU—ACORD pentru care se găsește corespondent în FISIER—PERSONAL. Întrucît s-a precizat că numele, prenumele și marca apar o singură dată* s-a utilizat cîmpul V care la prima scriere are valoarea 0 iar după aceea *1*. Pe ramura 'DA' a testului V—IVK=1 se trece o în V—IVK; în loc de V—MARCA tebuie scris V—NUMAR—MARCA.

13.46 *Î.* Scrieți procedura COBOL corespunzătoare schemei logice din fig. 13.17.

R.
```
INCEPUT.
       MOVE O   TO T—CISTIG—INTREPRINDERE
                   NUMĂR — CURENT
                   V — EOF
       OPEN INPUT FISIER—LUCRU—ACORD
            OUTPUT LISTA
       DISPLAY SPACES
       DISPLAY     'SITUATIA CISTIGURILOR IN ACORD'
       DISPLAY '. . . . . . . . . . . . . . . . . . . . . .'
       DISPLAY SPACES
READ FISIER—LUCRU—ACORD AT END MOVE 1 TO V—EOF.
       PERFORM MODUL—SECTIE UNTIL V—EOF = 1
       DISPLAY 'TOTAL INTREPRINDERE = ' T—CISTIG—
                                   INTREPRINDERE
       CLOSE FISIER—LUCRU—ACORD
             LISTA
       STOP—RUN.
```

* Ar fi lipsit de sens, ca pentru fiecate articcl care indică ce a executat în timp de o lună un om al muncii să se imprime numele, prenumele și marca (ele sint aceleași).

NUMAR	NUME SI PRENUME	FUNCTIA	CATEGORIE INCADRARE	RETRIBUTIE TARIFARA
n	a - n	n	n	n
4	40	1	1	4

Fig. 13.23.

SECTIE	LOC MUNCA	NUMAR MARCA	DOCUMENT		CANTITATE EXECUTATA	TARIF PE BUCATA
			NR	DATA		
n	n	n	n	n	n	n
2	4	4	5	4	5	2

Fig. 13.24.

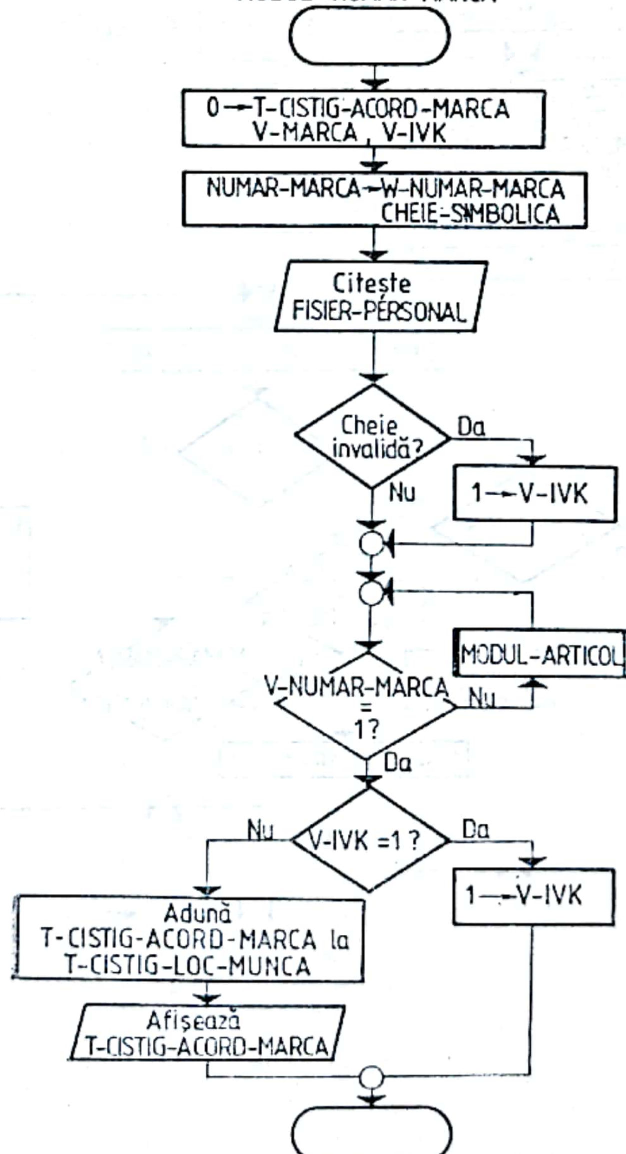


Fig. 13.25.

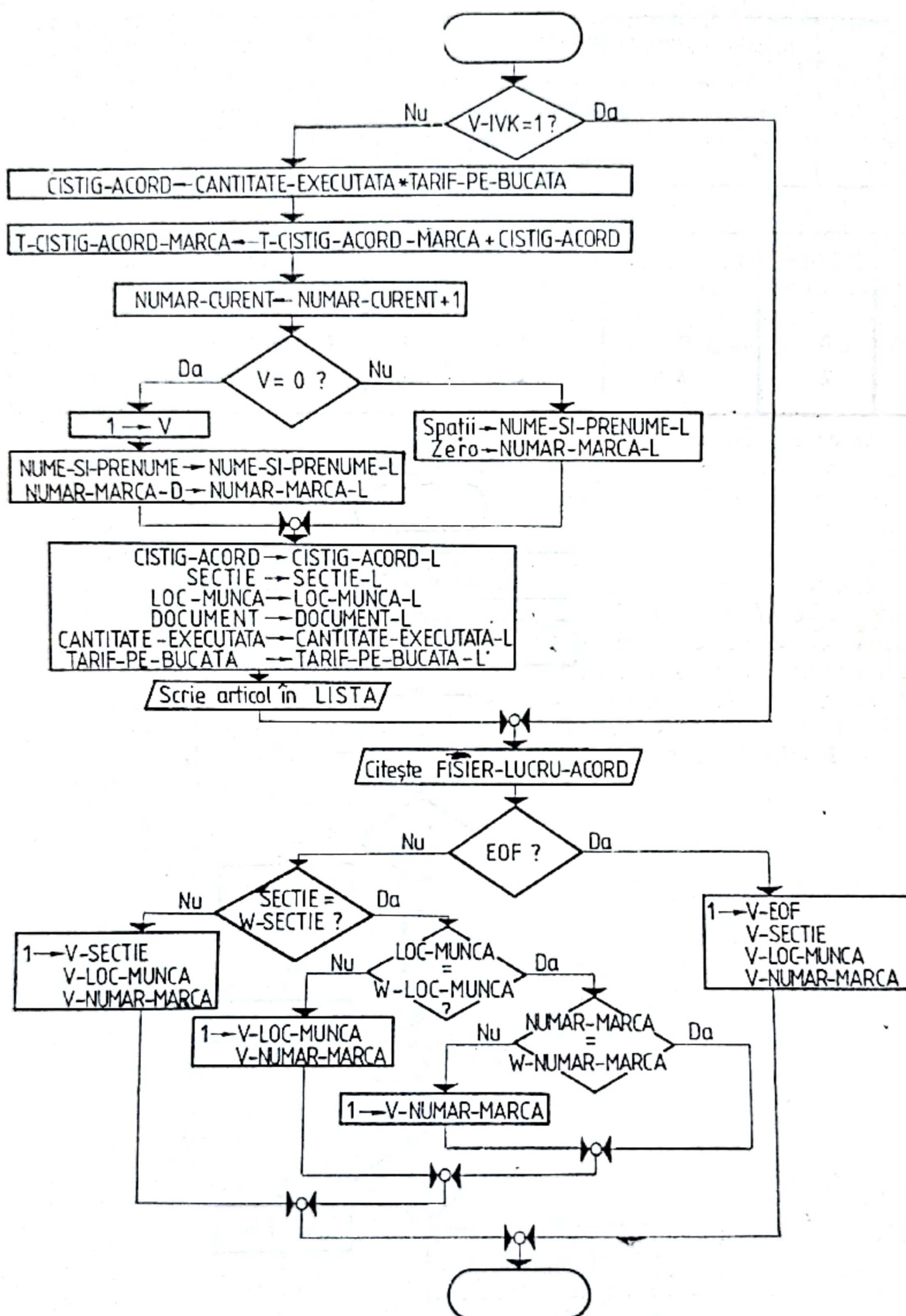

V-IVK=1?
Nu
Da
CISTIG-ACORD ← CANTITATE-EXECUTATA * TARIF-PE-BUCATA
T-CISTIG-ACORD-MARCA ← T-CISTIG-ACORD-MARCA + CISTIG-ACORD
NUMAR-CURENT ← NUMAR-CURENT + 1
V = 0 ?
Da
Nu
1 → V
Spatii → NUME-SI-PRENUME-L
Zero → NUMAR-MARCA-L
NUME-SI-PRENUME → NUME-SI-PRENUME-L
NUMAR-MARCA-D → NUMAR-MARCA-L
CISTIG-ACORD → CISTIG-ACORD-L
SECTIE → SECTIE-L
LOC-MUNCA → LOC-MUNCA-L
DOCUMENT → DOCUMENT-L
CANTITATE-EXECUTATA → CANTITATE-EXECUTATA-L
TARIF-PE-BUCATA → TARIF-PE-BUCATA-L
Scrie articol în LISTA
Citeşte FISIER-LUCRU-ACORD
EOF ?
Nu
Da
SECTIE = W-SECTIE ?
Nu
Da
1 → V-EOF
V-SECTIE
V-LOC-MUNCA
V-NUMAR-MARCA
1 → V-SECTIE
V-LOC-MUNCA
V-NUMAR-MARCA
LOC-MUNCA = W-LOC-MUNCA ?
Nu
Da
1 → V-LOC-MUNCA
V-NUMAR-MARCA
NUMAR-MARCA = W-NUMAR-MARCA
Nu
Da
1 → V-NUMAR-MARCA

Fig. 13.26.

13.47 *Î.* Scrieți procedura COBOL Corespunzătoare schemei logice din figura 13.25.

R.

```
MODUL—NUMAR—MARCA.
    MOVE O TO T—CISTIG—ACORD—MARCA, V—NUMĂR—
    MARCA  MOVE NUMAR—MARCA  TO W—NUMAR—
                MARCA CHEIE—SIMBOLICA
    READ FISIER—PERSONAL INVALID KEY
                       MOVE 1 TO V—IVK.
    PERFORM MODUL—ARTICOL UNTIL V—NUMĂR—
    MARCA = 1
    IF V—IVK = 1 MOVE 0 TO V—IVK
    DISPLAY 'ARTICOL INEXISTENT' CHEIE—SIMBO-
    LICA
    ELSE  ADD  T—CISTIG—ACORD—MARCA  TO T—CIS-
    TIG—LOC—MUNCA
    DISPLAY 'TOTAL CISTIG =' T—CISTIG  —ACORD—
    MARCA.
```

13.48 *Î.* Scrieți procedura COBOL corespunzătoare schemei logice din fig. 13.26.

R.

```
MODUL—ARTICOL.
   IF V—IVK =1 GO TO A ELSE
   MULTIPLY CANTITATE—EXECUTATA BY TARIF—PE—
   BUCATA
            GIVING CISTIG—ACORD
   ADD CISTIG—ACORD  TO  T—CISTIG—ACORD—MARCA
   ADD 1 TO  NUMAR—CURENT
   IF V = 0     MOVE  SPACES  TO  NUME—PRENUME—L
                MOVE ZEROS TO NUMAR—MARCA— L
      ELSE MOVE NUME—PRENUME TO  NUME—PRE-
      NUME—L
      MOVE  NUMĂR—MARCA  TO  NUMAR—MARCA—L.
      MOVE  CISTIG—ACORD  TO CISTIG—ACORD -L
      MOVE  SECTIE           TO SECTIE—L
      MOVE  DOCUMENT         TO DOCUMENT—L
      MOVE  CANTITATE -EXECUTATA  TO  CANTITATE—
      EXECUTATA—L
      MOVE TARIF—PE—BUCATA TO TARIF —PE—BUCATA
      —L
      WRITE  ARTICOL—LISTA  AFTER  1.
```

```
A. READ FISIER—LUCRU—ACORD AT END
                        MOVE 1 TO V—EOF
                                  V—SECTIE
                                  V—LOC—MUNCA
                                  V—NUMAR—MARCA
        GO TO REVENIRE
    IF SECŢIE=W—SECŢIE
                      MOVE 1= TO V—SECŢIE
                                 V—LOC—MUNCA
                                 V—NUMĂR—MARCA
    ELSE IF LOC—MUNCA NOT = W—LOC—MUNCA
                     MOVE 1 TO V—LOC—MUNCA
                               V—NUMAR—MARCA
    ELSE IF NUMAR—MARCA NOT = W—NUMAR—MARCA
                     MOVE 1 TO V—NUMAR—MARCA.
REVENIRE.
    EXIT.
```

13.49 Aşa cum se poate observa din răspunsul dat la întrebarea 13.45.I. programarea structurată nu exclude utilizarea instrucţiunii GO TO; ea presupune însă o utilizare raţională a acesteia. Instrucţiunea PERFORM permite modularizarea programelor şi crează premisele unei standardizări şi autodocumentări *.

13.50 *I.* Întrucît programarea structurată nu a găsit încă o manieră unitară de structurare a programelor încercaţi să întocmiţi o altă variantă de schemă logică pentru obţinerea situaţiei prezentată în fig. 13.14 utilizînd în intrare un singur fişier, FISIER—LUCRU—ACORD.

R. (fig. 13.27, 13.28, 13.29, 13.30, 13.31)
Structura din fig. 13.29 presupune utilizarea instrucţiuni PERFORM pentru apelul modulului ierarhic inferior

---

13.51 *I.* Prezentaţi pe scurt deficienţele variantei 1 de descompunere a modulului MODUL—PRELUCRARE (fig. 13.28)

. . . .
. . . .
. . . .
. . . .

---

* Spunem că un program este autodocumentat dacă prezintă suficiente explicaţii în legătură cu conţinutul şi poate fi înţeles cu uşurinţă de către o altă persoană în afara autorului. Sugerăm cititorului să însoţească orice paragraf, secţiune de comentarii în aşa fel încît programele să fie accesibile şi altor programatori.

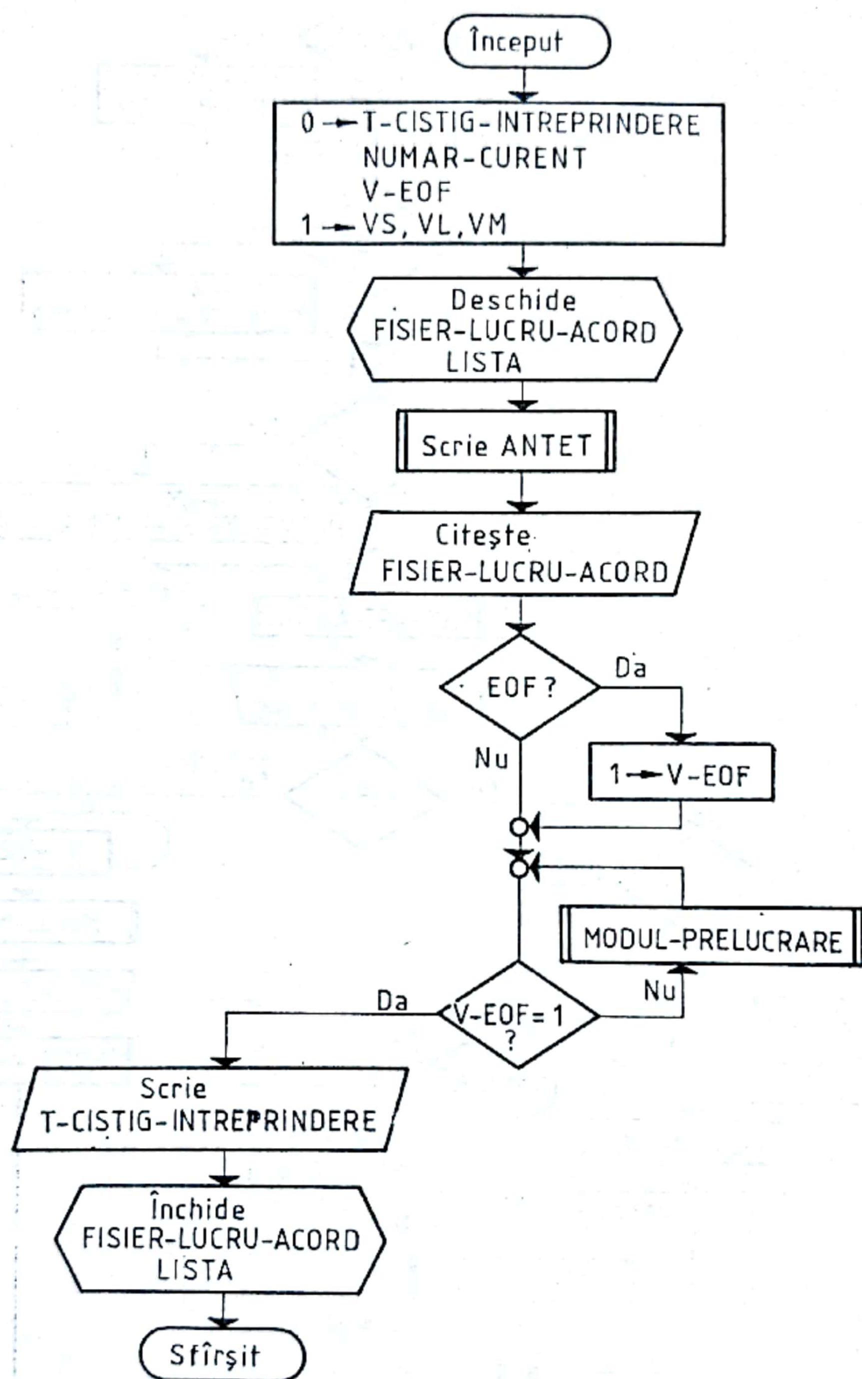

Început
0 → T-CISTIG-INTREPRINDERE
NUMAR-CURENT
V-EOF
1 → VS, VL, VM
Deschide
FISIER-LUCRU-ACORD
LISTA
Scrie ANTET
Citeşte
FISIER-LUCRU-ACORD
EOF ?
Da
Nu
1 → V-EOF
MODUL-PRELUCRARE
Nu
Da
V-EOF = 1
?
Scrie
T-CISTIG-INTREPRINDERE
Închide
FISIER-LUCRU-ACORD
LISTA
Sfîrşit

Fig. 13.27.

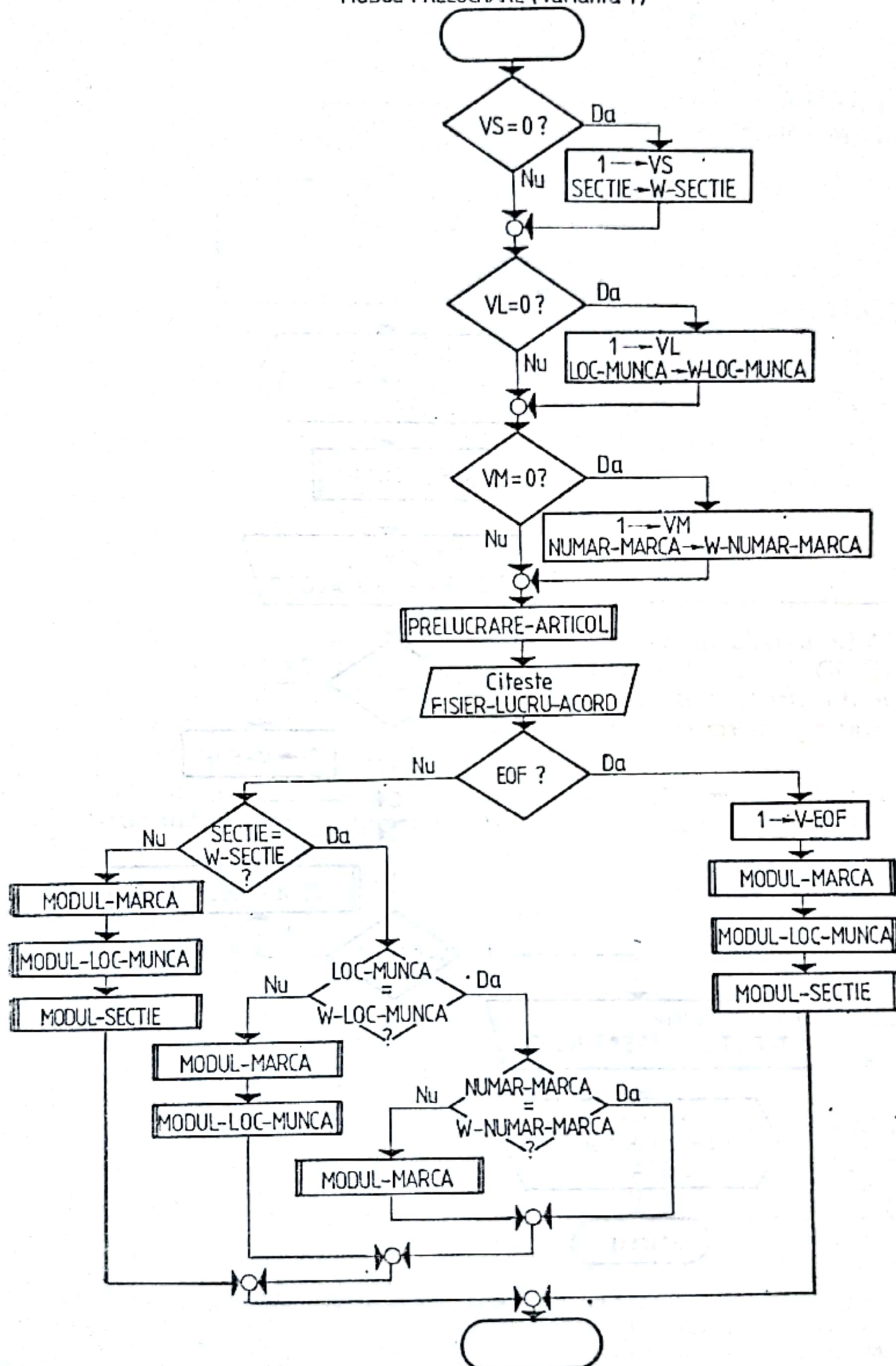

MODUL-PRELUCRARE (varianta 1)
VS=0?
Da
Nu
1→VS
SECTIE→W-SECTIE
VL=0?
Da
Nu
1→VL
LOC-MUNCA→W-LOC-MUNCA
VM=0?
Da
Nu
1→VM
NUMAR-MARCA→W-NUMAR-MARCA
PRELUCRARE-ARTICOL
Citeste
FISIER-LUCRU-ACORD
EOF ?
Nu
Da
SECTIE=
W-SECTIE
?
Nu
Da
1→V-EOF
MODUL-MARCA
MODUL-LOC-MUNCA
MODUL-SECTIE
LOC-MUNCA
=
W-LOC-MUNCA
?
Nu
Da
NUMAR-MARCA
=
W-NUMAR-MARCA
?
Nu
Da

Fig. 13.28.

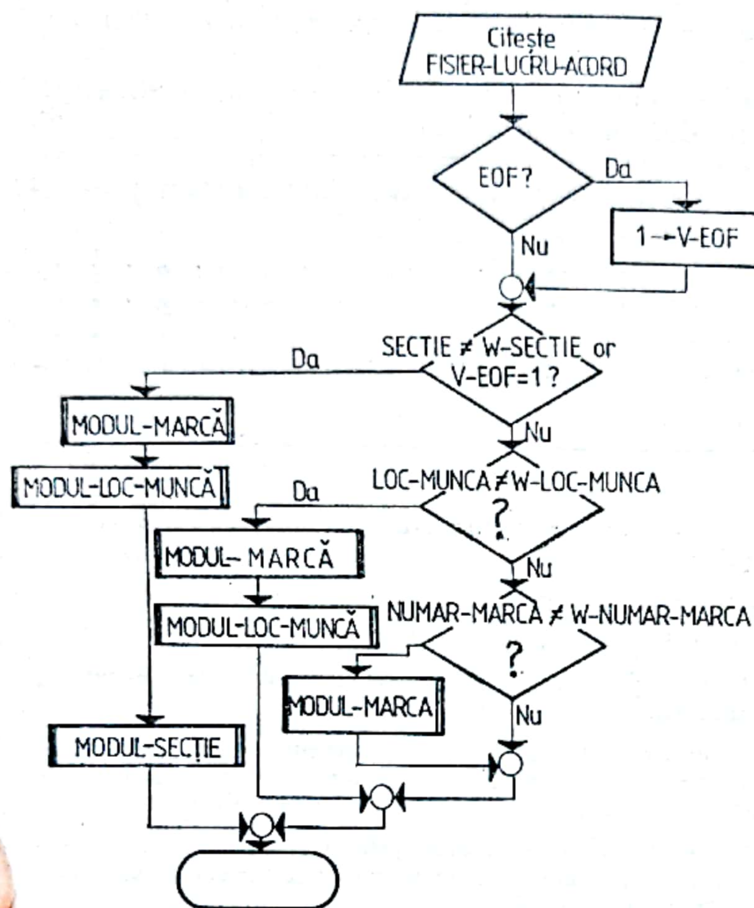


Fig. 13.29.

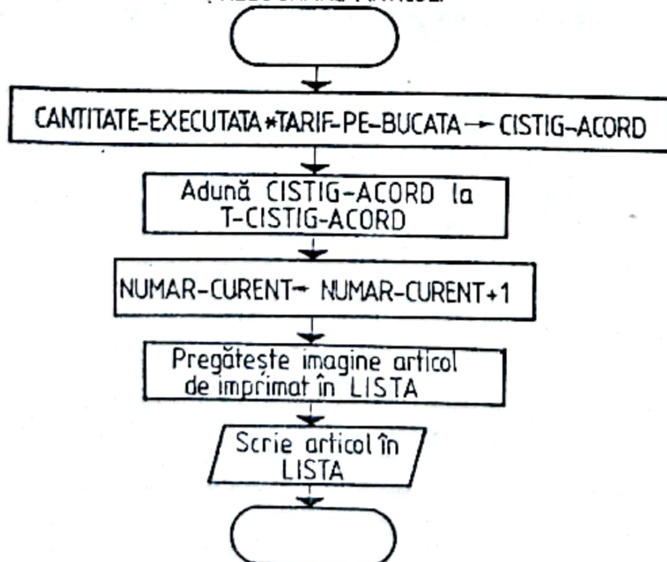


Fig. 13.30.

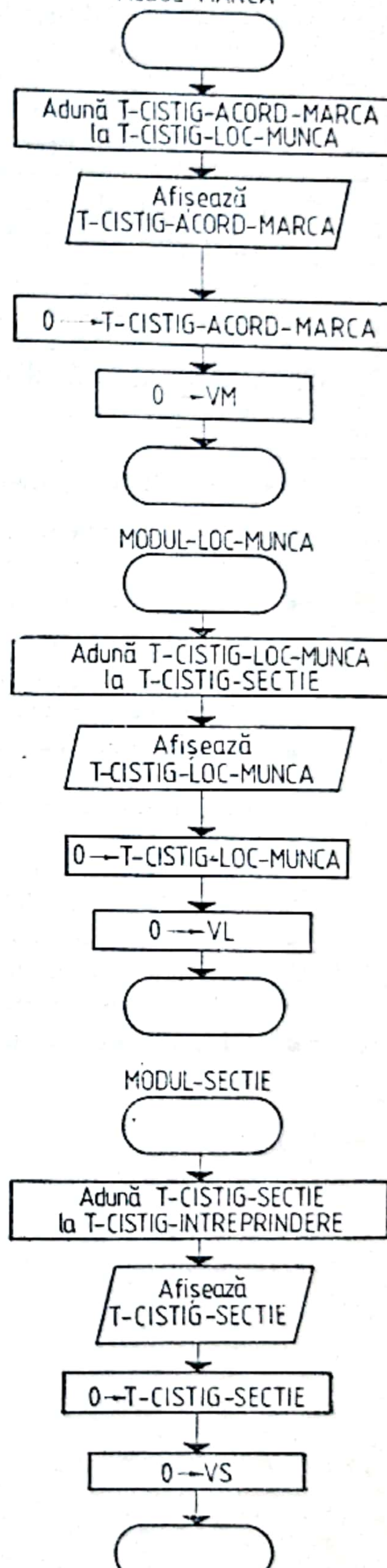


Fig. 13.31.

R. menţine redundanţa în conceperea programului în sensul că anumite module sînt apelate de mai multe ori; o combinare a condiţiilor de apel ar putea reduce substanţial numărul acestora (vezi fig. 13.29)
— nu redă intr o manieră clară relaţiile existente între indicatorii de grupare precizaţi în definirea problemei.

13.52 *I.* Care sînt, după părerea dumneavoastră, avantajele variantei prezentate în fig. 13.29?

. . . .

R. reduce numărul operaţiilor de apel de proceduri şi simplifică structura programului.

Reguli de stil în programare

- Înainte de scrierea programelor pentru fiecare dintre acestea se adoptă o mulţime standard de cuvinte — utilizator.
- Părţile logice ale cuvintelor trebuie separate prin liniuţa de unire.
- Identificatorii aleşi să fie cît mai sugestivi.
- Numele de paragrafe vor fi scrise pe linii separate şi vor fi însoţite de comentarii, în vederea unei mai mari accesibilităţi.
- Complexitatea unui program creşte odată cu numărul de niveluri de includere folosite în structura sa. Pentru simplficare aliniaţi corect instrucţiunile şi limitaţă extinderea în adîncime la trei niveluri de includere şi la cel mult o pagină imprimată.
- În orice paragraf, a cărui execuţie este controlată prin PERFORM VARYING pentru a descrie structuri repetitive se recomandă să nu se modifice valoarea variabilei de control în nici un fel prin instrucţiuri COBOL.
- Dacă o dată apare în mai multe articole diferite nu este obligatoriu ca acesteia să i se atribuie nume diferite. În acest caz, pentru referiri corecte în diviziunea PROCEDURE identificatorii care se repetă trebuie calificaţi folosind cuvintele rezervate IN sau OF.
- La modularizarea unui program aplicaţi principiile abstractizării şi compoziţiei.

LECȚIA a 14-a **SECȚIUNEA WORKING-STORAGE ȘI EDITAREA RAPOARTELOR**

- structura secțiunii WORKING—STORAGE
- variante de editare a rapoartelor
- cazuri practice

14.1 Datele organizate în fișiere se descriu în secțiunea FILE din diviziunea DATA. După caz, în program se pot utiliza și date care au un caracter intermediar în procesul de prelucrare și nu sînt cuprinse în fișiere. Datele intermediare sînt descrise în program într-o secțiune aparte numită WORKING—STORAGE (memorie de lucru).

14.2 Structura generală a secțiunii WORKING—STORAGE este următoarea :

[**WORKING—STORAGE SECTION.**
{rubrică de descriere a datelor.} ...]

În secțiunea WORKING—STORAGE se descriu datele independente folosind rubrici de nivel 77, 88 și grupate folosind rubrici de nivel 01—49, 88. Rubricile de nivel 77 trebuie să le preceadă pe cele de nivel 01—49.

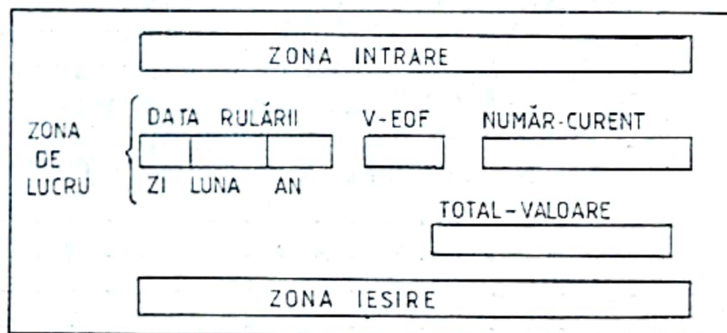


Fig. 14.1.

Prezentăm în continuare rubricile secțiunii WORKING—STORAGE pentru descrierea cîmpurilor din fig. 14.1.

```
WORKING—STORAGE SECTION.
77   NUMĂR—CURENT        PIC 9(4) VALUE 0.
77   TOTAL—VALOARE       PIC 9(7)V99 VALUE ZEROS.
01   DATA—RULĂRII.
     02  AN              PIC 99.
     02  LUNA            PIC 99.
     02  ZI              PIC 99.
```

14.3 *Î.* Scrieți instrucțiunea COBOL de preluare prin program a datei rulării de la sistemul de operare.

a) . . . .
. . . .

Dar dacă data se introduce cu ajutorul unei cartele?
b) . . . .
R. a) ACCEPT DATA—RULĂRII FROM DATE
b) ACCEPT DATA—RULĂRII

14.4 Pentru a păstra o legătură mai strînsă între descrierea datelor independente și rolul jucat de către acestea în program sugerăm cititorului să evite folosirea numărului de nivel 77 și să le grupeze în funcție de destinație și nivel ierarhic.

Spre exemplu, toate datele intermediare utilizate la nivelul unui modul se pot subordona unei rubrici de nivel 01:

```
01  DATE—MODUL—COMANDA.
    02  V—EOF                  PIC 9 VALUE 0.
        88  SFIRSIT—FIŞIER     VALUE 1.
    02  TOTAL—VALOARE          PIC 9(7)V99 VALUE 0.
    02  NUMĂR—CURENT           PIC 9(4) VALUE 0.
    02  DATA—RULĂRII.
        03  AN                 PIC 99.
        03  LUNA               PIC 99.
        03  ZI                 PIC 99.
```

14.5 *Î.* Într-un program, la nivelul modulului de prelucrare intitulat MODUL—GESTIUNE se folosesc cîmpurile de date intermediare TOTAL—VALOARE—GESTIUNE de natură numerică și lungime 9 + 2, V—GESTIUNE de natură numerică și lungime 1 și CONTOR—ARTICOLE de natură numerică și lungime 5. Cîmpul V—GESTIUNE are în program rolul variabilei de control și prin valoarea *1* desemnează sfîrșitul unei gestiuni. Cum se poate realiza descrierea acestora în secțiunea WORKING—STORAGE?

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

R. *Varianta a*

```
77  TOTAL—VALOARE—GESTIUNE   PIC 9(9)V99  VALUE 0.
77  V—GESTIUNE               PIC 9  VALUE 0.
    88  SFIRSIT—GESTIUNE.             VALUE 1.
77  CONTOR—ARTICOLE          PIC 9(5) VALUE 0.
```

Varianta b

```
01  VARIABILE—GESTIUNE
    02  TOTAL—VALOARE—GESTIUNE PIC 9(9)V99 VALUE 0.
    02  V—GESTIUNE             PIC 9 VALUE 0.
        88  SFIRSIT—GESTIUNE        VALUE 1.
02  CONTOR—ARTICOLE            PIC  9(5) VALUE 0.
```

14.6 *Î.* Dacă în secțiunea FILE clauza VALUE se utilizează numai în cazul datelor . . . . descrise cu numărul de nivel . . . . în secțiunea WORKING—STORAGE ea poate apare în orice rubrică de descriere . . . .

R. — nume de condiție
— 88
— a datelor elementare

14.7 În condițiile folosirii limbajului COBOL situațiile la imprimantă (rapoartele) se pot edita în una din variantele:

a) utilizînd un fișier în ieșire asignat la perifericul standard de ieșire (SYSOUT) și folosind instrucțiunea WRITE pentru scrierea rîndurilor;

b) utilizînd instrucțiunea de afișare DISPLAY fără a descrie în program un fișier la imprimantă;

c) folosind editorul COBOL și prin urmare secțiunea REPORT și instrucțiunile adecvate de editare.

Întrucît *varianta c* va fi prezentată pe larg în lecția a 17-a, vom insista în cele ce urmează asupra primelor două.

14.8 Programul 14.1 traduce în instrucțiuni COBOL algoritmul de citire a articolelor din F—BANDA și de editare a raportului prezentat în figura 14.2 în varianta utilizării instrucțiunii DISPLAY.

14.9 În practica programării, pentru rapoarte mai simple, se poate aplica o variantă combinată în care rîndul de detaliu este imprimat prin WRITE iar rîndurile de total și respectiv de antet prin instrucțiunea DISPLAY.

14.10 Editarea rapoartelor în varianta *b* impune analiza acestora în vederea determinării structurii rîndurilor de antet, de total și respectiv de detaliu.

14.11 În cele ce urmează prezentăm un astfel de program care editează la imprimantă raportul intitulat: „LISTA DE INVENTAR" folosind în intrare un fișier pe cartele numit FCINV (fig. 14.3 și 14.4).



Fig. 14.3.

Pentru fiecare articol din fișierul FCINV se scrie cîte un rînd numit și *rînd detaliu.*

14.12 În condițiile în care raportul ar cuprinde numai rîndurile de detaliu programul ar fi asemănător celui prezentat în paragraful 14.8 cu deosebirea că în acest caz se definește un fișier la imprimantă și se folosește pentru imprimare instrucțiunea WRITE.

14.13 *Î.* Scrieți rubrica FD și rubricile de descriere a datelor pentru fișierul FCINV și articolul asociat acestuia. (formatul de memorare: CANT 4+2, PRET 3+2, UM 3α).

. . . .
. . . .
. . . .

LISTA DE INVENTAR

DATA: ZZ/19ZZ

NRC	COD MAT	DENUMIRE	CANT	PRET	VALOARE
		TOTAL GENERAL			Z(7)9.99

Fig. 14.2.

```
ID DIVISION.
PROGRAM-ID. UP141.
    :
DATA DIVISION
FD F-BANDA RECORDING F
           LABEL RECORD STANDARD
           BLOCK CONTAINS 35 RECORDS.
01  ARTICOL.
02  COD-MAT          PIC          9(6).
    :
02  PRET             PIC          9(5)   V99   COMP-3.
WORKING-STORAGE      SECTION.
01  ARTICOL-DE-IMPRIMAT.
02  NUMAR-CURENT     PIC          ZZ9.
    :
02  VALOARE          PIC          Z(8)9.99.
01  DATE-GRUP.
02  NRC              PIC          9(4)   VALUE Ø.
    :
PROCEDURE DIVISION.
MODUL-PRINCIPAL.
    OPEN INPUT F-BANDA
    DISPLY SPACES
    DISPLAY    \ LISTA DE INVENTAR  /
    DISPLAY  \ ===============  /
    ACCEPT DATA-SISTEM
    MOVE  ZI  TO  ZI-L
    MOVE  LUNA  TO  LUNA-L
    MOVE  ANUL  TO  ANUL-L
    DISPLAY  SPACES
    DISPLAY  '       DATA                  '  DATA-EDITARII
    DISPLAY     SPACES
    DISPLAY  ' . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .'
    DISPLAY  ' = NRC = COD.MAT = DENUMIRE = CANT = PRET = VALOARE
    DISPLAY  ' . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .'
    READ F-BANDA AT END MOVE 1 TO SFF.
    PERFORM MODUL-OPERATIONAL UNTIL SFF=1
    DISPLAY 'TOTAL GENERAL =' TOTAL-GEN
    :
MODUL-OPERATIONAL.
    :
    DISPLAY ARTICOL-DE-IMPRIMAT
    READ F-BANDA AT END MOVE 1 TO SFF.
```

Programul 14.1.

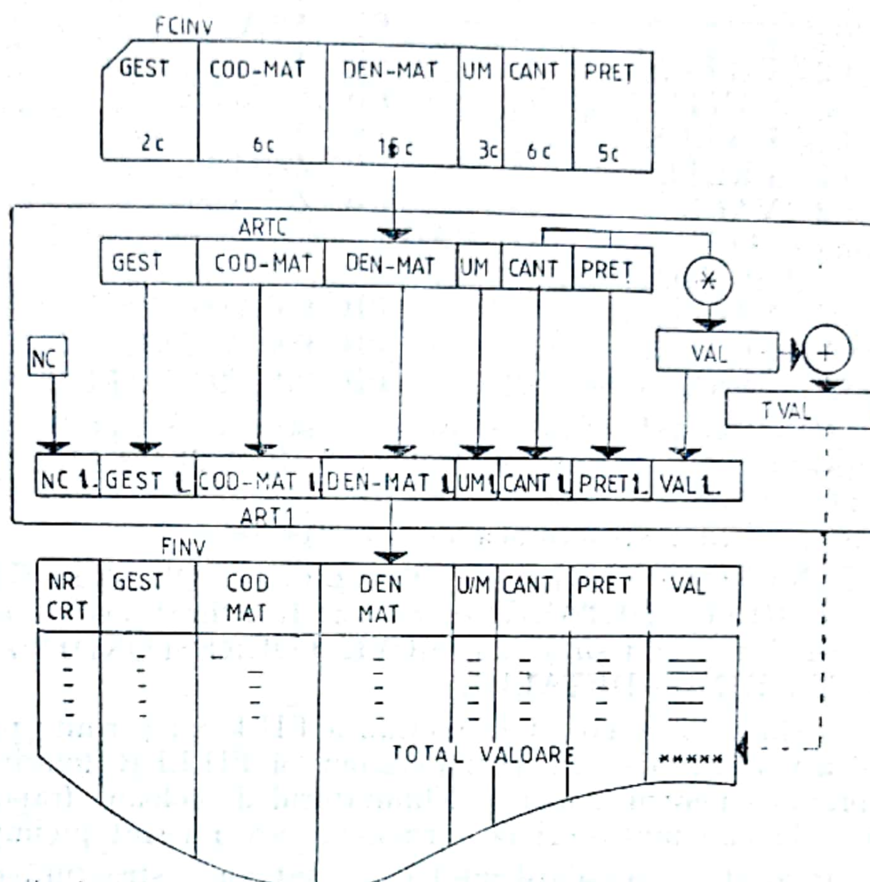


Fig. 14.4.

```
R.  FD  FCINV   RECORDING F
                LABEL RECORD OXITTED.
    01  ART—CARTELA.
        02  GEST                  PIC 99.
        02  COD—MAT               PIC 9(6).
        02  DEN—MAT               PIC X(15).
        02  UM                    PIC AAA.
        02  CANT                  PIC 9(4)V99.
        02  PRET                  PIC 9(3)V99.
```

14.14 Fişierul la imprimantă se poate descrie astfel:

```
    FD  FINV   RECORDING F
                          LABEL RECORD OMITTED.
    01  RÎND—DETALIU.
        02  FILLER                PIC X(5).
        02  NCL                   PIC ZZ9.
        02  GESTL                 PIC 99.
        02  FILLER                PIC 99.
        02  FILLER                PIC X.
        02  COD—MATL              PIC 9(6).
        02  DEN—MATL              PIC X(15).
        02  FILLER                PIC X.
```

```
        02  UML                   PIC AAA.
        02  FILLER                PIC X.
        02  CANTL                 PIC ZZZ9.99.
        02  FILLER                PIC X.
        02  PRETL                 PIC ZZ9.99.
        02  VALL                  PIC Z(5)9.99.
În secţiunea WORKING—STORAGE.
    01  GRUP--MODUL.
        02  VAL                   PIC 9(6)V99.
        02  NC                    PIC 999 VALUE 0.
        02  TVAL                  PIC 9(8)V99 VALUE 0.
```

14.15 Pentru a simplifica referenţierea datelor în program s-a preferat folosirea unor identificatori diferiţi cum ar fi PRET, PRETL; CANT, CANTL; DEN—MATL ş.a.m.d.

În aceste condiţii instrucţiunea de transfer se scrie:

MOVE CANT TO CANTL în loc de MOVE CANT OF ART—CARTELA TO CANT OF RÎND—DETALIU (dacă s-ar fi utilizat aceiaşi identificatori şi pentru zona de ieşire) sau MOVE CORRESPONDING ART—CARTELA TO RÎND—DETALIU.

Descrierea rîndului de editat în secţiunea FILE nu permite programatorului să dea valori iniţiale în special cîmpurilor FILLER, folosite în mod curent în practică pentru a marca delimitatorul de coloană (rapoartele se editează de cele mai multe ori pe formulare care nu sînt preimprimate).

14.16 Întrucît rapoartele (standardizate) de editat au o structură complexă ele trebuie analizate în vederea delimitării următoarelor categorii de rînduri:

— rînduri de antet de situaţie;
— rînduri de antet de pagină;
— rînduri de detaliu;
— rînduri de total;
— etc.

Pentru simplificarea exemplului primele două categorii de rînduri au fost numite rînduri de antet (fără să se mai specifice şi situaţia sau pagina).

14.17 *Î.* Pe exemplul din figura 14.5 precizaţi care sînt rîndurile de antet . . . .
rîndul de detaliu . . . . şi rîndul de total . . . . .

R. R1, R2, R3, R4, R5, R3; RINDC; RINDT

14.18 Modelul situaţiei la imprimantă este stabilit, de regulă, de către programator. El trebuie să respecte cerinţele de redare clară a rezultatelor prelucrării pentru a fi citite uşor de către utilizator. Mai trebuie reţinut că anumite rînduri se repetă în situaţie. Tipul de rînd respectiv va fi descris o singură dată şi imprimat ori de cîte ori este nevoie. În exemplul prezentat în paragraful 14.17 tipul de rînd RIND3 se va imprima de două ori, iar tipul de rînd RINDC (rînd detaliu) se va imprima de *n* ori.

14.19 *Î.* Vă rugăm să precizaţi de cîte ori va fi imprimat rîndul RINDT. . . . .

R. o singură dată

14.20 Rapoartele de editat cuprind, în general, mai multe tipuri de rînduri. Fiecare tip de rînd este descris în secțiunea WORKING—STORAGE folosind numerele de nivel, de la 01 la 49; inițializarea unor cîmpuri din structura acestor rînduri se realizează folosind clauza VALUE în rubricile de descriere a datelor.

```
R1     INTREPRINDEREA DIDACTICA

R2                   LISTA DE INVENTAR

R3     **************************************************************
R4     *NR *GE* COD  *      DEN MAT       *U/M*   CANT   * PRET * VAL   *
R5     *CRT*ST* MAT  *                    *   *          *      *       *
R3     **************************************************************

RINDC  zz9 99 999999 xxxxxxxxxxxxxxx  xxx zzzz9.99 zz9.99 zzzzz9.99




RINDT                                TOTAL VALOARE zzzzzzz9.99
```

Fig. 14.5.

În secțiunea FILE, fișierului la imprimantă (asignat perifericului standard în ieșire—SYSOUT) i se asociază o zonă de ieșire a cărei lungime este egală cu lungimea rîndului de lungime maximă descris în WORKING—STORAGE.

14.21 Pentru exemplul prezentat în paragraful 14.17 partea care se referă la descrierea datelor va avea următoarea structură:

```
DATA DIVISION.
FILE SECTION.
-----------------------------------------------------------------
FD FIINV RECORDING F LABEL RECORD OMITTED.
01  ARTI PIC X(66).
WORKING—STORAGE SECTION.
01  RIND1 . . . . . . . . . .
01  RIND2 . . . . . . . . . .
01  RIND3.
. . . . . . . . . . . . . . .
01  RIND4.
. . . . . . . . . . . . . . .
01  RIND5.
. . . . . . . . . . . . . . . .
01  RINDC.
. . . . . . . . . . . . . . .
01  RINDT.
. . . . . . . . . . . . . . .
```

14.22 Articolele descrise în WORKING—STORAGE SECTION se vor imprima după ce au fost transferate în zona articol a fișierului la imprimantă corespunzătoare descrierii generale a articolului în FILE SECTION. Transferul se poate realiza folosind instrucțiunea MOVE sau clauza FROM din

instrucţiunea WRITE (cu precizarea pregătirii prealabile a imaginii articolului descris în WORKING—STORAGE).

Pentru cazul discutat imaginea memoriei va fi următoarea (fig. 14.6):

14.23 *Î.* Fie ART—1 numele unui grup de date din W—S *. Scrieţi instrucţiunea WRITE pentru editarea conţinutului zonei ART—1 ştiind că fişierului la imprimantă i se asociază zona de ieşire numită RIND.

. . . .

R. WRITE RIND FROM ART—1 AFTER 1

14.24 Programul 14.2 descrie modelul situaţiei de la paragraful 14.17.

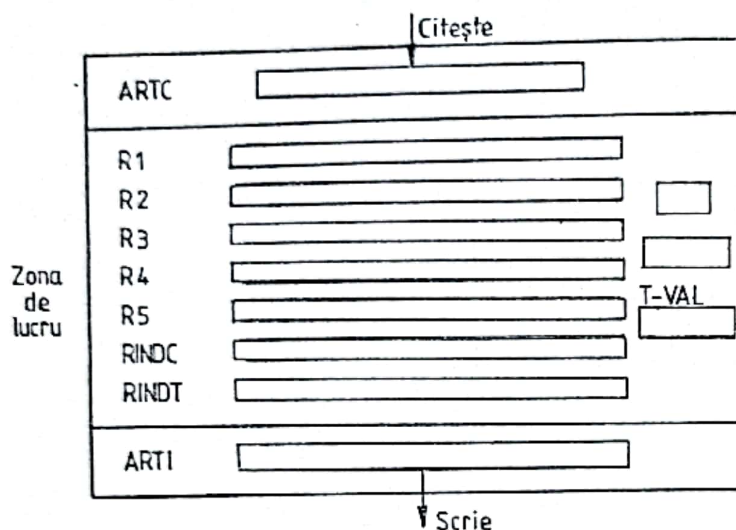


Fig. 14.6.

La editarea unui raport se are în vedere realizarea automată (prin program) a operaţiilor de gestionare a rîndurilor în cadrul paginii şi respectiv raportului. În acest scop programele de editare a rapoartelor în varianta b cuprind proceduri speciale cu funcţie de încadrare în pagină a rîndurilor de imprimat.

14.25 La proiectarea şi realizarea acestor proceduri avem în vedere:

— numărul maxim de rînduri de editat într-o pagină;

— variabila folosită pentru calcularea numărului de rînduri scrise într-o pagină;

— detectarea momentului cînd s-a ajuns la numărul limită de rînduri scrise în pagină şi efectuarea saltului la pagina următoare;

— precizarea operaţiilor de realizat la început de pagină.

Procedura de control a modului de încadrare în pagină poate avea structura:

```
1.  adună     1 la CONTOR
2.  dacă      CONTOR > n
              atunci
                      2.1. salt la pagina următoare
                      2.2. pregăteşte şi scrie antet pagină
                      2.3. atribuie o variabilei CONTOR
              altfel (*)
    sf—dacă
```

unde:

n, numărul maxim de rînduri de scris într-o pagină (în afara rîndurilor de antet pagină).

---

* W—S, iniţialele de la WORKING—STORAGE

UNITATEA
CONTUL

LUCRAREA | PROGRAMUL | INSTRUCȚIUNI DE PERFORARE | SIMBOLUL | PAG. DIN
PROGRAMATOR | DATA | SE PERFOREAZĂ | 73 IDENTIF. 80

SECVENȚA | CONTINUARE | INSTRUCȚIUNILE PROGRAMULUI

```
010 DATA DIVISION.
020 FILE SECTION.
030 ---------------------------
040 FD  FINV RECORDING V
050            LABEL RECORD OMITTED.
060 01  RIND                PIC X(66).
070 WORKING-STORAGE SECTION.
080 77  VAL                 PIC 9(6)V99.
090 77  NC                  PIC 999 VALUE 0.
100 77  T-VAL               PIC 9(8)V99 VALUE 0.
110 01  R1.
120     02 FILLER           PIC X(5) VALUE SPACES.
130     02 INTR             PIC X(26)
140                 VALUE  'INTREPRINDEREA DIDACTICA'.
150     02 FILLER           PIC X(35) VALUE SPACES.
160 01  R2.
170     02 FILLER           PIC X(19) VALUE SPACES.
180     02 ANTET            PIC X(20)
190                 VALUE  'LISTA DE INVENTAR '.
200     02 FILLER           PIC X(37) VALUE SPACES.
210 01  R3.
220     02 FILLER           PIC X(5) VALUE SPACES.
230     02 ASTERIX          PIC X(61) VALUE ALL '*'.
240 01  R4.
250     02 FILLER           PIC X(5) VALUE SPACES.
```

Programul 14.2.

UNITATEA
CONTUL

LUCRAREA	PROGRAMUL	INSTRUCŢIUNI DE PERFORARE	SIMBOLUL	PAG. DIN
PROGRAMATOR	DATA		SE PERFOREAZĂ	73 IDENTIF. 80

SECVENTA | CONTINUARE | INSTRUCTIUNILE PROGRAMULUI

```
010    02 FILLER          PIC X(61) VALUE '*NR *GE* COD   *   DEN MAT
020-     '     *U/M*   CANT  * PRET  *    VAL     *'.
030  01  R5.
040    02 FILLER          PIC X(5) VALUE SPACES.
050    02 FILLER          PIC X(61) VALUE '*CRT*ST* MAT    *
060-            *        *    *       *           *'.
070  01  RINDC.
080    02 FILLER          PIC X(6) VALUE SPACES.
090    02 NC-L            PIC ZZ9.
100    02 FILLER          PIC X VALUE SPACE.
110    02 GEST-L          PIC 99.
120    02 FILLER          PIC X VALUE SPACE.
130    02 COD-MAT-L       PIC 9(6).
140    02 FILLER          PIC X VALUE SPACE.
150    02 DEN-MAT-L       PIC X(15).
160    02 FILLER          PIC X VALUE SPACE.
170    02 UM-L            PIC XXX.
180    02 FILLER          PIC X VALUE SPACE.
190    02 CANT-L          PIC Z(4)9V,99.
200    02 FILLER          PIC X VALUE SPACE.
210    02 PRET-L          PIC ZZ9V,99.
220    02 VALOARE-L       PIC BZ(5)9V,99.
230  01  RINDT.
240    02 FILLER          PIC X(40) VALUE SPACES.
         :
```

14.26 În program se va urmări ca instrucţiunea de scriere a unui articol să fie precedată de instrucţiunile (sau de instrucţiunea de apel a acestora) care realizează controlul în vederea corectei încadrări în pagină.

14.27 Controlul deplasării hîrtiei se realizează cu ajutorul *benzii pilot*, care conţine perforaţii în canalele 0—7 corespunzătoare numărului de rînduri cu care se realizează avansarea hîrtiei. Deplasarea benzii pilot se realizează concomitent cu deplasarea hîrtiei şi corespunde cu unitatea standard *pagină*. În mod obişnuit pe o pagină se imprimă 66 rînduri.

14.28 Imprimanta poate deci realiza operaţii de editare propriu-zisă şi de avansare a hîrtiei cu un anumit număr de rînduri. Codul de salt asociat articolelor pentru realizarea unei corecte încadrări în pagină poate lua valorile din tabelul 14.1. Avansarea hîrtiei are loc numai dacă se montează banda pilot pe dispozitivul care controlează avansul hîrtiei (operaţie ce trebuie cerută în mod expres de către programator).

Tabelul 14.1

Cod de salt (hexazecimal)	Semnificaţie
E0	Avans interzis înainte şi după imprimare
C0	Avans interzis înainte de imprimare
C1	Avans un rînd înainte de imprimare
C2	2
C3	3
C4	4
C5	5
C6	6
C7	Avans { 7 } rînduri înainte de imprimare
C8	8
C9	9
CA	10
CB	11
CC	12
CD	13
CE	14
CF	15
F0	Avans definit de canalul 0 al benzii pilot
F1	1
F2	2
F3	3
F4	Avans definit de canalul { 4 } al benzii pilot
F5	5
F6	6
F7	7

14.29 ∂. În secţiunea WORKING—STORAGE se descriu datele de . . . . care pot fi . . . . sau . . . . datele independente se descriu cu numărele de nivel . . . . .

iar cele grupate cu numerele de nivel . . . .
În WORKING—STORAGE SECTION, dacă avem date independente și grupate, descrierea începe cu cele . . . .
Atunci cînd situația la imprimantă are mai multe tipuri de rînduri, descrierea rîndului general se face în . . . .
iar descrierea fiecărui tip de rînd se face în . . . .
Pentru a aduce conținutul unui un tip de rînd în zona—articol—ieșire în vederea imprimării, se va utiliza instrucțiunea WRITE ca opțiunea . . . .

R. — lucru
— independente
— grupate
— 77, 88
— 01—49,88
— independente
— FILE SECTION
— WORKING—STORAGE SECTION
— FROM

14.30 *Î.* Să se scrie secțiunea WORKING—STORAGE pentru
— cîmpurile independente — CODMAT format din 6 caractere;
— TOT—GEN cîmp de cumul format din 8 + 2 caractere;
— CONTSINT format din 3 caractere;
— cîmpul grup DATA—ZILEI format din cîmpurile ZI, LUNA, AN

. . . .
. . . .
. . . .
. . . .

Precizați erorile din liniile 140, 190 pag. 221 și 60 pag. 222.

R.
```
WORKING—STORAGE SECTION.
77 CODMAT PIC 9(6).
77 TO—GEN PIC 9(8)V99 VALUE 0.
77 CONTSINT PIC 999.
01 DATA—ZILEI.
   02 ZI PIC 99.
   02 LUNA PIC 99.
   02 AN PIC 99.
```
Nu s-a marcat apostroful în coloanele 27,54 (linia 140): 27 linia 190; 12 linia 60 și punctul terminale în linia 140.

14.31 Într-o a treia variantă descrierea (prin enunțuri COBOL) și editarea rapoartelor se poate realiza folosind editorul COBOL (vezi lecția 17).

14.32 În practică, descrierea situațiilor la imprimantă se face în mod riguros folosind în acest sens formulare preimprimate care au în vedere structura unei pagini la imprimantă și folosirea benzii pilot.

LECȚIA a 15-a ORGANIZAREA ȘI PRELUCRAREA DATELOR TIP TABLOU

- structuri de tablou
- clauza OCCURS
- PERFORM VARYING
- SET
- SEARCH

15.1 Un tablou este o structură omogenă, ce cuprinde unul sau mai multe cîmpuri de aceeași natură și lungime care apar succesiv în cadrul unui articol sau a unui cîmp grup. Organizarea acestora în tablouri ușurează munca programatorului care nu mai trebuie să repete de *n* ori descrierea aceluiași cîmp (unde *n* reprezintă numărul cîmpurilor elementare din tablou). De asemenea la proiectarea și realizarea programelor care utilizează structuri de tablou se cîștigă timp, prin utilizarea frecventă a structurii repetitive și a instrucțiunii PERFORM.

Fie cîmpurile:

```
CANTITATE-CONTR-1
CANTITATE-CONTR-2

CANTITATE-CONTR-12
```

care conțin cantitățile contractate pe cele 12 luni de către întreprinderea DIDACTICA. Să presupunem că toate cele 12 date sînt perforate într-o singură cartelă.

Structura articolului este următoarea:

```
01 ARTICOL.
   02 CANTITATE-CONTR-1        PIC 9(6).
   02 CANTITATE-CONTR-2        PIC 9(6).
   02 CANTITATE-CONTR-3        PIC 9(6).
. . . . . . . . . . . . . . . . . . . . . . . . . . . .
   02 CANTITATE-CONTR-12       PIC 9(6).
```

Pentru a elimina repetarea descrierii de date (ce au aceeași natură și lungime și apar succesiv în cadrul unui articol) se apelează la structura de tablou și la clauza OCCURS.

La descrierea unui tablou este necesar să se precizeze:

— natura datelor care reprezintă elementele tabloului (precizată prin clauza PICTURE);

— dimensiunile tabloului;

— numărul de elemente ce corespund fiecărei dimensiuni;

— după caz, cheile pentru ordonarea elementelor tabloului;
— indecşii asociaţi fiecărei dimensiuni;

Tablourile pot avea una, două sau trei dimensiuni.

Un indice reprezintă un cîmp de natură numerică, un literal numeric sau registrul special TALLY, iar *un index* un tip aparte de indice (valoarea unui index reprezintă adresa relativă a zonei rezervată unui cîmp din cadrul tabloului).

La prelucrarea datelor se pot utiliza două categorii de tablouri:
— tablouri cu număr fix de elemente;
— tablouri cu număr variabil de elemente.

Pentru cea de a doua categorie de tablouri se prevăd valori ce indică numărul de elemente din tablou.

15.2 Reluînd exemplul de mai sus, vom descrie articolul în varianta folosirii clauzei OCCURS:

```
01  ARTICOL.
    02  CANTITATE—CONTRACTATA          PIC 9(6)
                  OCCURS               12 TIMES.
```

La precizarea elementelor de mai sus (cu excepţia naturii datelor) se foloseşte clauza OCCURS care are următorul format simplificat:

OCCURS număr—întreg TIMES

unde:

• *număr—întreg* indică numărul de elemente ale tabloului.

• *Clauza OCCURS* poate apare, pentru descrierea datelor cu structură de tablou, la orice număr de nivel exceptînd 01 şi 77.

• *Opţiunea număr—întreg TIMES* se foloseşte la descrierea tablourilor cu un număr fix de elemente indicat prin număr—întreg. Folosirea clauzei OCCURS în rubrica de descriere a unei date indică faptul că data în cauză reprezintă elementele ce corespund unei dimensiuni a tabloului.

15.3 Pentru a permite identificarea celor *n* elemente ale tabloului se utilizează un *indice* încadrat între paranteze.

De exemplu construcţiile:

CANTITATE—CONTRACTATA (1)
CANTITATE—CONTRACTATA (12)

permit identificarea primei şi respectiv a celei de a 12-a cantităţi din tablou.

În varianta CANTITATE—CONTRACTATA (I) se identifică al i-lea element în funcţie de valoarea curentă a lui I.

În expresia CANTITATE (12) ca indice s-a folosit un literal numeric în timp ce în expresia CANTITATE (I) indicele reprezintă o dată numerică.

15.4 Scrierea în formularul de programare se face astfel:

|C|A|N|T|I|T|A|T|E|—|C|O|N|T|R|A|C|T|A|T|A| |(|I|)|

La scrierea în formularul de programare între numele tabloului şi paranteza deschisă trebuie să existe cel puţin un spaţiu; nici un spaţiu nu trebuie să urmeze după paranteza deschisă sau să preceadă paranteza închisă.

15.5 În procesul prelucrării datelor pot apare şi cazuri cînd elementele tabloului sînt succesiuni de date elementare de diverse tipuri.

Exemplu:

```
01    ARTICOL.
   02    COD—MATERIAL                              PIC 9(6).
   02    CONTRACT                    OCCURS 4 TIMES.
      03    NUMĂR—DOCUMENT                         PIC 9(5).
      03    CANTITATEA—CONTRACTATA                 PIC 9(5).
```

Articolul va conţine pentru fiecare fel de material numărul documentului şi cantitatea contractată pe fiecare trimestru.

Putem identifica elementele tabloului descris mai sus utilizînd construcţiile :

CONTRACT (3)

sau :

NUMĂR—DOCUMENT (3)
CANTITATE—CONTRACTATA (3)

15.6 În cazul în care indicii folosiţi la identificarea elementelor dintr-un tablou sînt cîmpuri de natură numerică, ei trebuie descrişi în WORKING—STORAGE SECTION folosind o rubrică de date cu structura :

02 nume—indice *PICTURE* S9(5)
[USAGE *COMPUTATIONAL*].

Opţiunea COMPUTATIONAL din cadrul clauzei USAGE permite memorarea datelor în format binar ceea ce, aşa cum am văzut în lecţia 1, duce la creşterea performanţelor în prelucrare (prin ocuparea unui spaţiu de memorare cît mai mic şi prin evitarea operaţiilor de conversie necesare în procesul de prelucrare).

15.7 Cînd pentru identificarea elementelor unui tablou se foloseşte varianta indecşilor, aceştia nu trebuie descrişi prin rubrici aparte deoarece compilatorul le asociază implicit cîte o zonă de 4 locaţii. Dacă un index a fost asociat unei dimensiuni, în mod obligatoriu el trebuie utilizat pentru identificarea elementelor respectivei dimensiuni.

Formatul simplificat al clauzei OCCURS pentru descrierea unui tablou în varianta folosirii indecşilor este :

OCCURS număr—întreg TIMES
INDEXED BY {nume—index} ... .

În cadrul unui program care operează cu structuri de tablou, datelor index li se atribuie valori cu ajutorul instrucţiunilor COBOL de prelucrare a tablourilor SET, SEARCH, PERFORM VARYING (vezi paragrafele următoare) ca şi cum ar reprezenta indici.

Exemplu de folosire a datelor index:

```
01  TABLOU—CU—DOUĂ—DIMENSIUNI.
    02  LINIE   OCCURS  10
                           INDEXED BY I.
        03  ELEMENT   OCCURS 10 PIC 9(5)V99
                      INDEXED BY J.
```

unde:
I, referenţiază cele 10 linii ale tabloului;
J, referenţiază cele 10 elemente din cadrul liniei I.

15.8 În varianta folosirii indicilor tabloul de mai sus are următoarea descriere:

```
01 TABLOU—CU—DOUĂ—DIMENSIUNI.
   02 LINIE OCCURS 10.
      03 ELEMENT OCCURS 10 PIC 9(5)V99.
```

iar separat se descriu indicii:

```
01 ZONA—INDICI.
   02 I           PIC 99  VALUE 0.
   02 J           PIC 99  VALUE 0.
```

Î. Prezentaţi pe scurt avantajele indexării tablourilor . . . . . .
. . . .

R. Indexarea tablourilor scuteşte programatorul COBOL de a mai descrie variabilele de adresare a elementelor în vederea prelucrării.

15.9 Pentru a da valori iniţiale unui tablou, în practică, se poate face apel la clauzele VALUE şi REDEFINES.

Exemplu:

```
02 LUNILE—ANULUI.
   03 FILLER PIC X(10) VALUE 'IANUARIE'.
   03 FILLER PIC X(10) VALUE 'FEBRUARIE'.
   03 FILLER PIC X(10) VALUE 'MARTIE'.
   03 FILLER PIC X(10) VALUE 'APRILIE'.
   03 FILLER PIC X(10) VALUE 'MAI'.
   03 FILLER PIC X(10) VALUE 'IUNIE'.
   03 FILLER PIC X(10) VALUE 'IULIE'.
   03 FILLER PIC X(10) VALUE 'AUGUST'.
   03 FILLER PIC X(10) VALUE 'SEPTEMBRIE'.
   03 FILLER PIC X(10) VALUE 'OCTOMBRIE'.
   03 FILLER PIC X(10) VALUE 'NOIEMBRIE'.
   03 FILLER PIC X(10) VALUE 'DECEMBRIE'.
02 TABLOU—LUNI REDEFINES LUNILE—ANULUI.
   03 LUNA              OCCURS 12
                        PIC X(10).
```

Cîmpul grup LUNILE—ANULUI a fost redefinit ca fiind un tablou cu o dimensiune. La folosirea clauzei OCCURS în corelaţie cu clauzele VALUE şi REDEFINES trebuie respectate restricţiile: elementelor unui tablou nu li se pot atribui valori iniţiale prin clauza VALUE; un tablou cu număr variabil de elemente nu poate fi redefinit.

15.10 Pentru a atribui cîmpului LUNA—DE—EDITAT valoarea „SEPTEMBRIE" se foloseşte comanda MOVE în structura:

```
MOVE LUNA (9) TO LUNA—DE—EDITAT
```

sau dacă în etapa precedentă i s-a atribuit lui I valoarea 9

```
MOVE LUNA (I) TO LUNA—DE—EDITAT
```

15.11 Programatorul COBOL poate folosi tablouri cu una, două şi trei dimensiuni.

Exemple:

- *tablou cu o dimensiune:*

```
02 SECŢIE       OCCURS  10
                PIC     9(7)V99.
```

- *tablou cu două dimensiuni:*

```
01  TABLOU
    02  SECTIE      OCCURS     10.
        03  ATELIER  OCCURS  10
                     PICTURE 9(7)V99.
```

- *tablou cu trei dimensiuni:*

```
01  TABLOU.
    02  SECTIE OCCURS 10.
        03  ATELIER OCCURS 10.
            04  SCHIMB    OCCURS 3
                          PICTURE 9(7)V99.
```

În exemplele de mai sus s-a descris, folosind structura de tablou, producţia realizată de către subunităţile unei întreprinderi în trei variante:

a) numai la nivel de secţie;

b) la nivel secţie şi atelier;

c) la nivel secţie, atelier şi schimb.

15.12 Identificarea elementelor din tablou se realizează în felul următor:

a) SECTIE (I)

b) ATELIER (I, J)

unde:

I, indică secţia,
J, atelierul din cadrul acesteia.
Pentru a referenţia producţia atelierului **4** din secţia **2** se scrie ATELIER (2 4)

c) SCHIMB (I, J, K)

unde:

I, desemnează secţia;
J, desemnează atelierul;
K, desemnează schimbul.
Pentru a referenţia producţia schimbului **2** din atelierul **3** şi secţia **5** se scrie:
SCHIMB (5 3 2)

15.13 În varianta folosirii indecşilor există două modalităţi de referire a elementelor unei dimensiuni şi anume:

a) *directă*, cînd se precizează indexul:

Exemplu:
SECTIE (I)
unde:
I, reprezintă indexul;

b) *relativă*, cînd indexul este urmat de un operator aritmetic (+ sau —) şi de un număr întreg:

Exemplu:
SECTIA (I+3).

15.14 *Î.* Din cele prezentate rezultă un al doilea format simplificat al clauzei OCCURS care este:

. . . .
. . . .

R. OCCURS între TIMES
INDEXED BY index—1.

15.15 Î. Descrieți un tablou cu 7 elemente care să aibă ca valori inițiale zilele săptămînii

. . . .

R.
```
01  ZILELE—SĂPTĂMÎNII.
    02  FILLER      PIC X(8)  VALUE   'LUNI'.
    02  FILLER      PIC X(8)  VALUE   'MARȚI'.
    02  FILLER      PIC X(8)  VALUE   'MIERCURI'.
    02  FILLER      PIC X(8)  VALUE   'JOI'.
    02  FILLER      PIC X(8)  VALUE   'VINERI'.
    02  FILLER      PIC X(8)  VALUE   'SÎMBĂTĂ'.
    02  FILLER      PIC X(8)  VALUE   'DUMINICA'.
01  TABLOU—ZILE  REDEFINES ZILELE—SĂPTĂMÎNII.
    02  ZIUA        OCCURS 7
                    PIC X(8).
```

15.16 Î. Scrieți instrucțiunea pentru transferul conținutului celui de al 5-lea element din TABLOU—ZILE în zona ZI—DE—EDITAT

. . . .

R. MOVE ZIUA (5) TO ZI—DE—EDITAT

15.17 Î. Scrieți în formularul de programare SCHIMBUL (I, J, K) și precizați regula de separare a indicilor

R. |S|C|H|I|M|B|U|L| |(|I|,| |J|,| |K|)|

sau

|S|C|H|I|M|B|U|L| |(|I| |J| |K|)|

15.18 Indicii care referențiază elementele unui tablou se separă printr-o virgulă și un spațiu sau cel puțin un spațiu.
Atunci cînd numărul elementelor din tablou este variabil (pentru tablouri cu o singură dimensiune) se poate utiliza clauza OCCURS în varianta precizării numărului minim și respectiv maxim de elemente.

Spre exemplu la descrierea unui tip de articol care cuantifică datele privind evidența personalului se va utiliza o dată de tip tablou privind situația copiilor; numărul de copii diferă de la o persoană la alta și poate lua valori între 0 și 15.

Descrierea datelor va fi următoarea:

```
01 ARTICOL—PERSONAL.
   02 NUMAR—MATRICOL      PIC 9(5).
   02 NUME—PRENUME        PIC A(30).
   02 FUNCTIA             PIC X(10).
```

```
02  NUMĂR—COPII                    PIC 99.
02  TABLOU—COPII                    OCCURS 0 TO 15
            DEPENDING ON NUMĂR—COPII.
    03  DATA—NAŞTERII              PIC 9(6).
    03  ÎNCADRAREA—SC              PIC 9.
```

Observație Cînd la descrierea unui tip de articol se folosește clauza OCCURS cu opțiunea DEPENDING ON data de tip tablou trebuie să fie descrisă ultima în cadrul articolului.

Din exemplul de mai sus deducem că pentru tablouri cu o singură dimensiune și număr variabil de elemente folosim următorul format al clauzei OCCURS:

OCCURS întreg—1 **TO** întreg—2 [**TIMES**]
DEPENDING ON nume—dată.

unde:

- *întreg—1*, desemnează limita inferioară (care poate fi și zero);
- *întreg—2*, desemnează limita superioară;
- *nume—dată*, desemnează cîmpul care conține numărul exact de elemente *întreg—1, întreg—2*.

Pentru a realiza ordonarea crescătoare sau descrescătoare a elementelor dintr-un tablou se folosesc opțiunile ASCENDING și respectiv DESCENDING în cadrul clauzei OCCURS.

15.19 Dacă în exemplul prezentat la paragraful 15.18 se cere ordonarea tabloului TABLOU—COPII în funcție de valorile crescătoare ale cîmpului DATA—NAŞTERII se realizează următoarea descriere:

```
01  ARTICOL—PERSONAL.
    02 NUMĂR—MATRICOL PIC 9(5).
    02 NUME—PRENUME PIC A(30).
    02 FUNCŢIA PIC X(10).
    02  NUMĂR—COPII                  PIC 99.
    02  TABLOU—COPII       OCCURS  0  TO  15
            DEPENDING ON NUMĂR—COPII
            ASCENDING KEY DATA—NAŞTERII.
        03  DATA—NAŞTERII            PIC 9(6).
        03  ÎNCADRAREA—SC            PIC 9.
```

15.20 Din exemplele prezentate rezultă următorul format general al clauzei OCCURS:

OCCURS { între—1 TIMES | între—2 **TO** întreg—3 TIMES **DEPENDING** ON nume—dată-1 }

[{ **ASCENDING** | **DESCENDING** } KEY IS nume—dată—2 [,nume—dată—3] . . .]

[**INDEXED** BY nume—index—1 [, nume—index—2] . . .].

unde:

- *opțiunea întreg—1 TIMES*, se folosește pentru descrierea tablourilor cu un număr fix de elemente;

• *opțiunea întreg—2 TO întreg—3 TIMES DEPENDING ON*, se folosește pentru descrierea tablourilor cu număr variabil de elemente;

• *opțiunile ASCENDING/DESCENDING*, se folosesc pentru a preciza ordonarea elementelor din tablou;

• *opțiunea INDEXED BY* se folosește pentru a preciza indecșii asociați dimensiunilor tabloului.

15.21 Pentru prelucrarea datelor organizate în tablouri cu o dimensiune se utilizează instrucțiunile:

PERFORM...VARYING, SEARCH și SET.

Dintre acestea frecvența de utilizare cea mai mare o deține instrucțiunea PERFORM VARYING și are următorul format simplificat:

PERFORM nume—prelucrare—1 [**THRU** nume—prelucrare—2]
VARYING *v* **FROM** *i* **BY** *p* **UNTIL** condiție

unde:

• *v*, variabilă folosită pentru adresarea elementelor din tablou și pentru realizarea operației de control;

• *i*, valoarea inițială a lui v;

• *p*, pasul sau valoarea adăugată lui v la fiecare iterație;

• *condiție*, o expresie condițională care permite controlul în vederea ieșirii din structura repetitivă.

15.22 Pentru a însuma producția realizată de către fiecare secție din cadrul intreprinderii DIDACTICA (vezi paragraful 15.11.) se folosește instrucțiunea PERFORM care trece controlul procedurii INSUMARE cu revenirea la instrucțiunea imediat următoare din program, cînd valoarea logică a condiției I>10 este adevărată:

```
PERFORM INSUMARE VARYING I FROM 1 BY 1
                              UNTIL I>10
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
INSUMARE.
    ADD SECTIA (I) TO TOTAL—PRODUCTIE—REALIZATA.
```

15.23 În prelucrarea datelor organizate în tablouri cu două dimensiuni se utilizează instrucțiunea PERFORM cu formatul:

PERFORM nume—prelucrare—1 [**THRU** nume—prelucrare—2]
VARYING v_1 **FROM** i_1 **BY** p_1 **UNTIL** condiție—1
AFTER v_2 **FROM** i_2 **BY** p_2 **UNTIL** condiție—2

unde:

• v_1, v_1, reprezintă variabilele de control pentru cele două dimensiuni (linie, coloană);

• p_1, p_2, valorile de incrementare a variabilelor v_1 și respectiv v_2;

• *condiție—1*, *condiție—2*, reprezintă condiții simple sau compuse prin a căror testare se stabilește ieșirea din structura repetitivă (din ciclu).

Logica executării instrucțiunii PERFORM, în acest format, este redată în figura 15.1.

Care sînt erorile ce apar în schemă?
Analizați îndeosebi inițializarea variabiei v_2.

Valorile curente ale variabilelor v_2 şi v_1 pot fi utilizate pentru adresarea elementelor tablourilor.

15.24 Pentru tablourile cu 3 dimensiuni se mai adaugă încă un aliniat cu opţiunea AFTER în structura :

AFTER v_3 FROM i_3 BY p_3 UNTIL condiţie—3

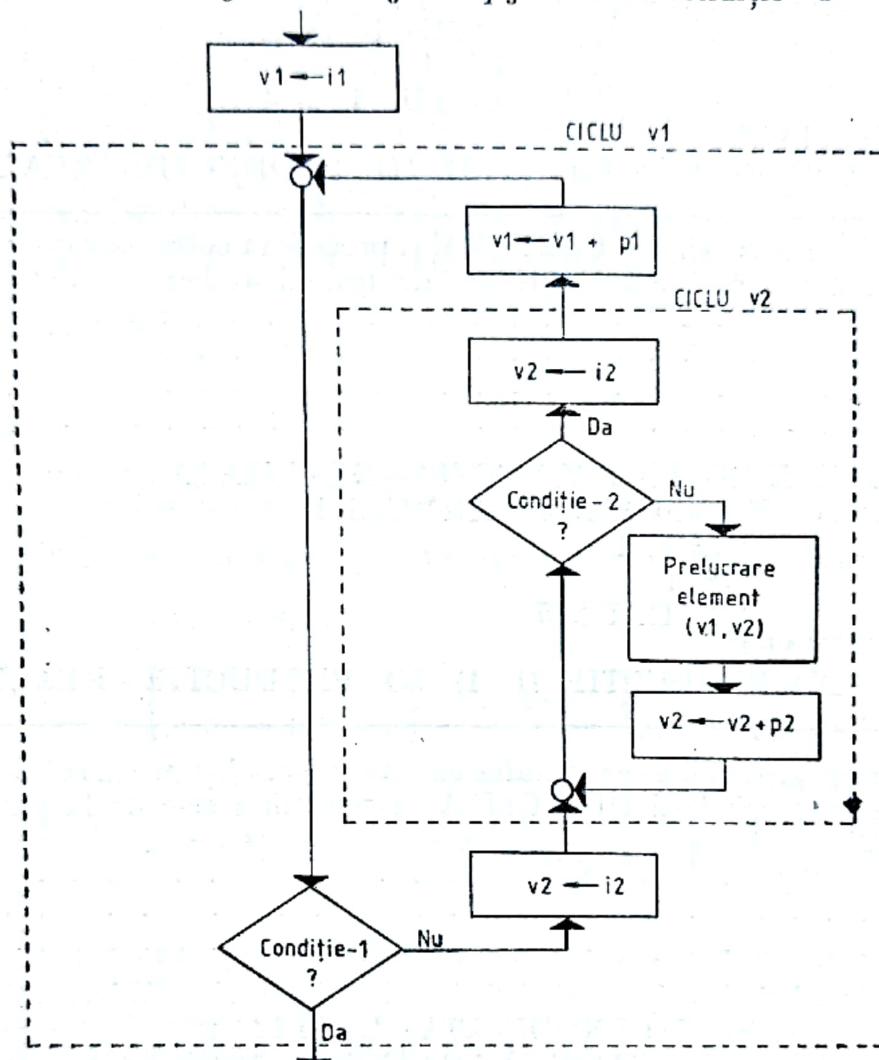


Fig. 15.1.

15.25 *?.* Formulaţi instrucţiunea PERFORM şi procedura asociată pentru însumarea elementelor de pe linia a 5-a din tabloul cu două dimensiuni descris mai jos.

ATELIER / SECTIE	ATELIER 1	ATELIER 2	ATELIER 3	ATELIER 4
Secţia 1	34 500	35 000	33 000	31 000
Secţia 2	27 000	28 300	30 000	25 000
Secţia 3	21 000	20 000	18 000	30 000
Secţia 4	15 000	17 000	18 000	13 000
Secţia 5	25 000	35 000	17 000	15 000

. . . .

R.
```
MOVE ZERO TO PRODUCTIE—REALIZATA
PERFORM INSUMARE VARYING I
                     FORM 1
                     BY 1
                     UNTIL  I > 4
INSUMARE.
   ADD PRODUCTIE (5, I) TO  PRODUCTIE—REALIZATA.
```

15.26 *Î.* Scrieţi instrucţiunea PERFORM şi procedura corespunzătoare pentru însumarea producţiei obţinute de primul atelier din fiecare secţie

. . . .

R.
```
MOVE ZERO TO PRODUCTIA—REALIZATA
PERFORM INSUMARE VARYING I
             FROM 1
             BY 1
             UNTIL I >5
INSUMARE.
   ADD PRODUCTIE (I, 1) TO  PRODUCTIE—REALIZATA.
```

15.27 *Î.* Scrieţi procedura pentru aflarea producţiei obţinute de către atelierele întreprinderii DIDACTICA, datele fiind cele de la paragraful 15.25.

. . . .

R.
```
MOVE ZERO TO PRODUCTIA—REALIZATĂ
PERFORM INSUMARE VARYING I FROM 1
                  BY 1
                  UNTIL I >5
   AFTER J  FROM 1
            BY 1
            UNTIL I >4
. . . . . . . . . . . . . . . . . . . . . . . .
INSUMARE.
   ADD PRODUCTIE (I,  J) TO PRODUCTIE—REALIZATA.
```

15.28 *Instrucţiunea SET* realizează atribuirea unei anumite valori la unul sau mai mulţi indecşi.

Dacă, spre exemplu, descriem în varianta cu indexare tabloul de la paragraful 15.25.

```
01 TABLOU—PRODUCTIE—REALIZATA.
    02 SECTIE  OCCURS 5
               INDEXED BY I.
       03      ATELIER OCCURS 4
               PIC        9(5)
               INDEXED    BY J.
```

Instrucţiunea SET pentru a atribui indexul I valoarea 3 este:

```
SET   I   TO   3
```

Acelaşi efect se poate obţine prin setul de instrucţiuni:

```
MOVE 3   TO   ZONA—VALOARE
SET   I  TO   ZONA—VALOARE
```

15.29 Pentru mărirea cu o unitate a indexului I se foloseşte instrucţiunea SET în formatul SET .... UP BY ....

Exemplu:

```
SET I UP BY 1
```

unde:

- valoare lui I înainte de execuţia instrucţiunii este 3;
- valoare lui I după execuţia instrucţiunii este 4.

Din cele prezentate rezultă că instrucţiunea SET poate avea unul din formatele:

a) pentru atribuirea de valori

SET {index—1 | nume—data—1} ... **TO** {index—2 | nume—dată—2}

b) pentru mărirea sau micşorarea valorii unuia sau mai multor indecşi

SET {index—1} ... {**UP BY** | **DOWN BY**} {nume—dată | literal}

unde:

- UP ... BY, opţiune utilizată pentru mărirea valorii indexului;
- DOWN BY, opţiune utilizată pentru micşorarea valorii indexului.

Indiferent de formatul utilizat *nume—dată—1*, *nume—dătă—2* trebuie să desemneze cîmpuri elementare numerice iar valorile lor şi ale literalului să fie pozitive şi numere întregi. În operaţia de atribuire conform formatului 1 trebuie să fie implicat cel puţin un index.

15.30 *Instrucţiunea SEARCH* permite descrierea unor cicluri care au ca obiectiv căutarea unui element într-un tablou cu o dimensiune.

Căutarea se poate realiza:

A) — *secvenţial*, caz în care se foloseşte formatul

SEARCH nume—tablou
  [**VARYING** index]
  [**AT END** instrucţiunea—1 ...]
  WHEN condiţie—1 instrucţiunea—2
  [**WHEN** condiţie—2 instrucţiunea—3 ...].

unde:

• *nume—tablou* numele tabloului care apare într-o rubrică de descriere de date cu folosirea clauzelor OCCURS și INDEXED BY;

• *index* unul din indecșii asociați tabloului;

• *instrucțiunea—1* instrucțiunea sau secvența de instrucțiuni ce se execută în cazul în care elementul căutat nu este găsit în tablou (condiție—1, condiție—2 nu sînt niciodată îndeplinite);

• *instrucțiunea—2/ instrucțiunea—3* instrucțiunile sau secvențele de instrucțiuni executate în cazul în care elementul a fost găsit și prin urmare valoarea logică a condițiilor *condiție—1, condiție—2,* .... este adevărat; cînd în instrucțiunea SEARCH apar mai multe condiții ele sînt evaluate în ordinea în care ele apar în instrucțiune.

15.31 Dacă în instrucțiunea SEARCH nu apare clauza VARYING variabila de control a ciclului este primul index din rubrica de descriere a tabloului. În condițiile folosirii clauzei VARYING, prin program, variabilei de control index *i* se atribuie o valoare inițială folosind instrucțiunea SET.

În execuția instrucțiunii SEARCH există corespondență între *condiția—1/instrucțiunea—2* și respectiv *condiția—2/ instrucțiunea—3.*

Exemplu:
Fie tabloul PRODUCȚIE cu următoarea descriere:

```
01 PRODUCȚIE.
   02 PRODUCȚIE—SECTIE   PIC 9(7)
                         OCCURS 10
                         INDEXED BY I.
```

Pentru a afla secția care are producția mai mare de 30000 se scrie următoarea secvență COBOL:

```
SET I TO 1
SEARCH PRODUCȚIE—SECȚIE
       WHEN PRODUCTIE—SECTIE > 30000
       SET VARIABILA TO I
       MOVE 1 TO V
IF  V = 1
   THEN
     DISPLAY 'SECTIE CU PRODUCȚIE > 30000 INEX'
   ELSE
     DISPLAY 'SECTIA A =' VARIABILA
             'ARE PRODUCTIE > 30000'.
```

15.32 Prin urmare căutarea în tablou începe cu elementul identificat prin conținutul lui I (în cazul de față 1) și se termină cînd a fost găsit cel care satisface o anumită condiție (în cazul de față PRODUCTIE > 30000).

B) — *binar* folosind formatul:

```
SEARCH ALL
    [AT END instrucțiunea—1 ...].
     WHEN condiție instrucțiune—2 ....
```

Elementele tabloului care fac obiectul unei căutări binare trebuie să fie ordonate în funcție de valorile cheii (cheilor) precizate în rubrica de descriere a tabloului; prin urmare ele vor conține clauza OCCURS cu opțiunile INDEXED și ASCENDING/DESCENDING. În varianta căutării

binare procesul de căutare este iterativ iar relaţia de determinare a valorii indicelui i este :

$$i = \left[\frac{1 + n}{2}\right]$$

unde : n, numărul de elemente din tablou.

Căutarea binară este cunoscută în literatura de specialitate şi sub numele de căutare dicotomică ; la prima căutare fie că se identifică elementul în cauză fie că se precizează în care parte a tabloului este memorat.

Deci dacă :

— $t(i) = v$ elementul a fost găsit ;
— $t(i) < v$ elementul căutat se află în a doua parte a tabloului;
— $t(i) > v$ elementul căutat se află în prima parte a tabloului,

unde :

- v, valoare elementului căutat
- t, numele tabloului ;
- i, indicele de adresare.

Exemplu:

— *descrierea tabloului*

```
        01  CONSUM-MATERIALE.
            02  MATERIALE  OCCURS 20
                           INDEXED BY I
                           ASCENDING KEY COD-MAT.
                03  COD-MAT           PIC 9(7).
                03  CANTITATEA        PIC 9(5)V99.
```

— *căutarea în tablou* a materialului de cod 3456789

```
        SEARCH ALL  MATERIALE
                    AT END MOVE 1 TO V
                    WHEN COD-MATERIAL = 3456789
                    MOVE COD-MAT (I) TO COD-MAT-E
                    MOVE CANTITATEA (I) TO CANTITATEA-E
                    DISPLAY CONSUM.
   IF  V = 1
       THEN
         DISPLAY 'MATERIALUL DE COD 3456789 INEXISTENT'.
```

LECŢIA a 16-a TESTAREA ŞI DEPANAREA PROGRAMELOR

- fazele tratării unui program
- tipuri de erori
- testarea programelor
- instrucţiunea PRINT

16.1 Din lecţiile precedente a rezultat că pentru fiecare problemă supusă studiului se construieşte, în general, un program care în forma în care este scris pe formular nu poate fi direct executabil. Mai întîi, programul este perforat pe cartele şi se obţine ceea ce am numit formatul sursă (fig. 16.1.).

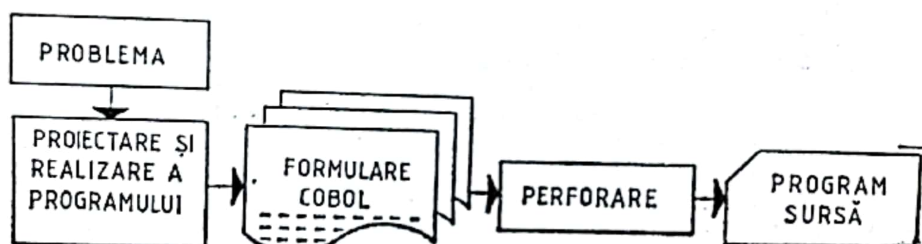


Fig. 16.1.

16.2 *Î.* Trecerea programului din format sursă în format obiect este realizată de către un program special al sistemului de operare, numit . . . . . .

R. Compilator

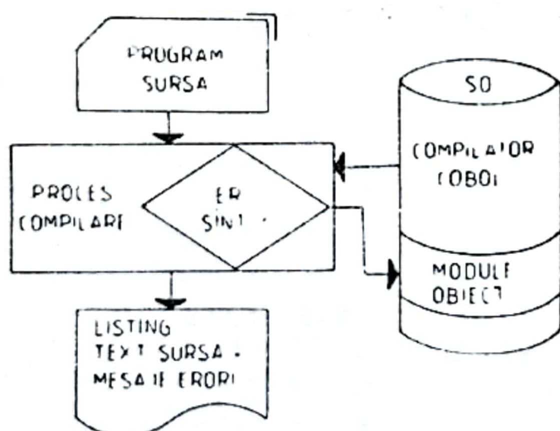
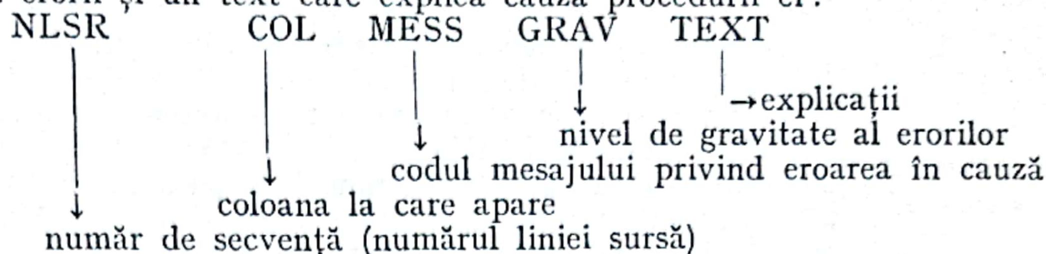


Fig. 16.2.

16.3 În procesul de compilare mai întîi se delimitează elementele care compun programul şi apoi se realizează analiza (pe grupuri de elemente) pentru a vedea dacă au fost respectate construcţiile componentelor limbajului de programare (regulile gramaticale). Dacă programul a fost formulat corect (din punct de vedere sintactic) se generează în mod automat modulele (programul) obiect (fig. 16.2.).

16.4 Textul sursă reprezintă exact programul care a fost perforat pe cartele. Fiecare linie sursă apare la un număr de ordine (care constituie şi mijlocul de identificare în caz de eroare).

16.5 Mesajul de eroare precizează : linia sursă la care a apărut eroarea, gravitatea erorii şi un text care explică cauza procedurii ei :



16.6 În funcţie de nivelul de gravitate, erorile care apar în faza de compilare se împart în :

a) *erori de grad 1,* cu rol de avertisment.

Exemplu:

ZONA–1 | 1 | 2 | 3 | 4 | 5 | ZONA–2 | 2 | 3 | 4 | 5 |

MOVE ZONA–1 TO ZONA–2

Deoarece ZONA–1 este mai mare decît ZONA–2, în procesul de transfer va avea loc o trunchiere a valorii; compilatorul avertizează programatorul de această eroare furnizînd un mesaj de nivel 1.

b) *erori de grad 2,* care semnalează sintaxa incorectă a unor construcţii din program. Compilatorul încearcă să corecteze programul în funcţie de anumite opţiuni. Spre exemplu dacă se declară 02 CANTITATE PIC 9(6) V 99 compilatorul menţionează descrierea eronată de dată sau lipsa punctului terminal.

c) *erori de grad 3,* sînt de gravitate ridicată, pentru care compilatorul nu încearcă să aducă corecţii. Dacă în program, compilatorul întîlneşte construcţia 02 PRETUL–UNITAR PIC 9(30) semnalează o eroare de nivel 3 (ştiut fiind că o dată numerică poate fi memorată pe maximum 18 locaţii).

d) *erori de grad 4,* erori foarte grave, care apar în cazuri speciale cînd se opreşte traducerea programului.

16.7 1. Menţionaţi erorile ce vor fi detectate de către compilatorul în următoarea secvenţă de program:

```
2 PROGRAM←ID. PROGRAM1.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
8 SELECT FISIER–DISC  ASSIGN D
                       ORGANIZATION INDEXED.
                       ACCESS SEQUENTIAL
                       RECORD KEY CODM.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
20  FD RECORDING F
       LABEL RECORD STANDARD.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
27  77  CONTOR PIC 9999VALUE 0.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
30  PROCEDURE DIVISION
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

40 WRITE ARTICOL INVALID KEY DISPLAY CHEIE ERONATA MOVE 1 T IVK.

R. Linia : 2, caracterul ← nu este permis în limbajul COBOL (trebuie înlocuit cu liniuţă de unire);
8, punctul se scrie o singură dată şi anume după ultima clauză a frazei SELECT;
20, după FD nu s-a trecut numele fişierului;
27, între ultimul caracter din şablon şi clauza VALUE trebuie lăsat cel puţin un spaţiu;
30, după numele diviziunii se trece caracterul "punct",
40, orice literal nenumeric trebuie să apară delimitat de caracterele „apostrof"

16.8 *Î.* Scrieţi corect aceleaşi elemente din secvenţa precedentă

. . . .
. . . .
. . . .
. . . .
. . . .

R.
```
2        PROGRAM-ID. PROGRAM1.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
8           SELECT FISIER-DISC ASSIGN D
                     ORGANIZATION INDEXED
                     ACCESS SEQUENTIAL
                     RECORD KEY CODM.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
20  FD      FISIER-DISC RECORDING F
               LABEL RECORD STANDARD.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
27  77      CONTOR PIC 9999 VALUE 0.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
30  PROCEDURE DIVISION.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
40          WRITE ARTICOL INVALID KEY
                     DISPLAY 'CHEIE ERONATA'
                     MOVE 1 TO IVK.
```

16.9 Dacă în listingul rezultat în urma compilării apar erori (în general cele de grad superior lui 1), înseamnă că trebuie să procedaţi la detectarea şi eliminarea lor, după care veţi recompila programul.

16.10 *Î.* Dacă programul nu mai conţine erori grave, compilatorul generează . . . .

16.11 *R.* programul obiect

Programul obiect este o sumă de module obiect, în format binar translatabil (BT) în care se includ, de obicei:

— *un modul de date,* care cuprinde datele descrise în secţiunea

WORKING—STORAGE, articolele asociate fișierelor tratate în mod MOVE etc;

— *unul sau mai multe module de date comune*, cîte un modul pentru fiecare fișier folosit în program (modulul de date comune conține deci zona articol asociată fișierului);

— *un modul program*, care cuprinde instrucțiunile din diviziunea de procedură (există posibilitatea împărțirii instrucțiunilor din diviziunea de procedură în mai multe module program).

16.12 Pentru a deveni executabile, modulele obiect trebuie unite într-un program (cu sau fără includerea unor module speciale cum ar fi modulele SGF). Spunem că se trece în etapa de *realizare a legăturilor* și se obține programul imagine memorie translatabilă (IMT), numit astfel deoarece adresele datelor și instrucțiunilor au semnificație numai în cadrul programului (nefiind însă asociate zonele reale ale memoriei).

Programul care realizează legăturile (editorul de legături) furnizează la sfîrșit o situație, care cuprinde structura programului obținut (pe segmente și module) și/sau erorile detectate. În acest moment programul poate

16.13 fi încărcat și lansat în execuție.

Erorile care apar în faza de editare a legăturilor sînt cauzate în general

16.14 de formularea eronată a cartelelor de comandă (vezi lecția 18.).

Î. Pentru a ajunge cu un program în faza de execuție trebuie să parcurgem fazele de . . . . . . și de . . . . în aceste faze se trece din formatul . . . . în formatul . . . și respectiv . . .

R. — compilare
— editare a legăturilor
— sursă; BT; IMT

16.15 În vederea prelucrării datelor, programul în format IMT este încărcat în memoria centrală (în totalitate sau pe segmente) și lansat în execuție.

Un program căruia i s-au eliminat erorile de compilare și respectiv de editare a legăturilor nu este în mod obligatoriu și un program bun (care permite obținerea rezultatelor dorite).

16.16 O primă posibilitate pe care o are programatorul pentru a vedea dacă un program este bun, este de a-l supune unor operații de testare folosind date adecvate. În acest fel spunem că detectăm erorile de semantică sau de logică. Numai în această fază ne putem da seama dacă s-a respectat întrutotul algortimul lucrării. Așa cum o să vedem în lecția 21 trebuiesc folosite metode de programare care să permită, încă din faza de proiectare, să se facă aprecieri asupra corectitudinii programului. Nu trebuie așteptată faza testării care poate duce la reluarea muncii de la capăt.

16.17 Erorile de semantică impun revizuirea proiectului de program (a schemei logice) pentru a vedea dacă s-au respectat întrutotul specificațiile problemei.

16.18 Pentru a realiza o testare completă a programului o atenție deosebită se va acorda alegerii datelor de testare, care trebuie făcută în strînsă legătură cu structura programului.

În vederea ușurării muncii de testare a programului trebuie urmărit ca programul să fie conceput și realizat cu maximă atenție folosind o metodologie unitară de lucru, în care se îmbină într-un tot unitar:

— tehnica modularizării programului;
— tehnica utilizării structurilor standard de control;
— tehnica autodocumentării programelor;
— controlul de calitate privind respectarea cu strictețe a algoritmului problemei.

16.19 *Î.* În procesul de tratare a unui program trebuie analizate și eliminate erorile de . . . . . care se referă la . . . . erorile de . . . . care se referă la . . . . erorile de . . . . care se referă la . . . .

R. — sintaxa;
— utilizarea eronată a construcțiilor specifice limbajului COBOL;
— editarea legăturilor;
— utilizarea eronată a comenzilor adresate editorului de legături;
— semantică;
— nerespectarea algoritmului lucrării sau compunerea necorespunzătoare a unor părți din program cu efect asupra ordinii de execuție.

16.20 Erorile ce apar în fiecare fază trebuie eliminate din program. Activitatea de depistare și eliminare a erorilor mai este cunoscută și sub numele de *depanare*.

16.21 *Î.* Reprezentați grafic etapele tratării unui program scoțînd în evidență rezultatele obținute.

R. (fig. 16.3.)

16.22 Preocupările actuale sînt îndreptate către noi metode și tehnici de programare care urmăresc îndeosebi eficientizarea muncii de programare prin introducerea pe cît posibil a unor structuri de bază în programare.

16 23 Pentru a facilita activitatea de depanare a programelor (cînd s-a ajuns în faza de testare) limbajul COBOL permite introducerea unor rutine de tratare a drumului parcurs, prin intermediul instrucțiunii *PRINT*, care are următorul format:

```
7    12
+    PRINT  [{nume—dată—1}] [[{nume—dată—2}]...]
             {literal—1  }    {literal—2  }
```

Programatorul va trebui să înștiințeze compilatorul de folosirea în program a instrucțiunii PRINT.

16.24 În momentul testării programului se va tipări la imprimantă numele paragrafului în care apare instrucțiunea PRINT, iar dacă opțiunile *nume—dată* și *literal—2* sînt utilizate și conținutul acestora.

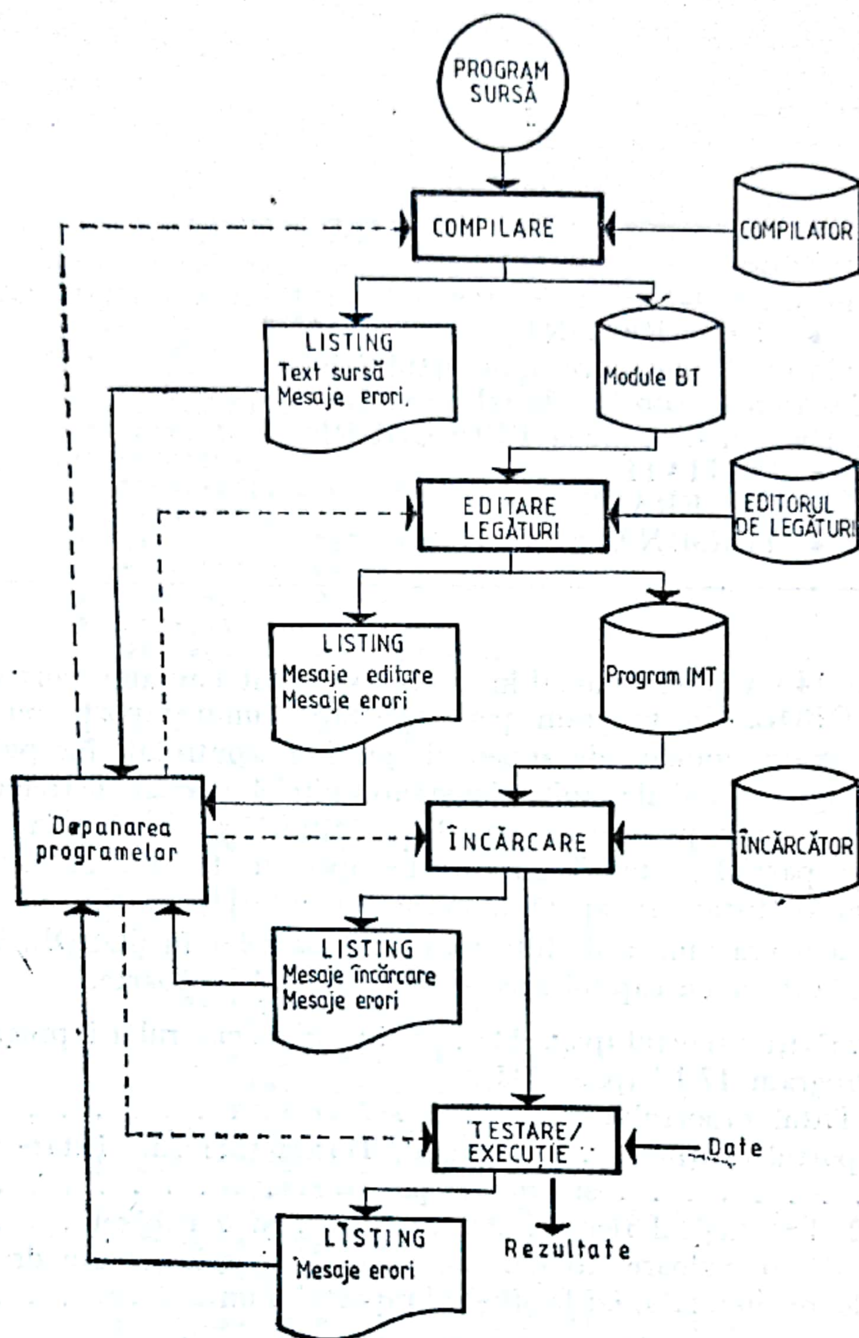


Fig. 16.3.

16.25 Programul după ce a fost testat poate fi inclus, într-o bibliotecă de program și utilizat la nevoie (vezi lecția 22).

Pentru verificarea cunoștințelor rezolvați testele 7 și 8, paginile 366, 371.

LECȚIA a 17-a EDITORUL DE RAPOARTE COBOL

- structura unui raport
- rubrica FD pentru descrierea fişierului asociat raportului
  - clauza REPORT
- rubrica de descriere a raportului RD
- descrierea grupelor de editare
- verbe din diviziunea PROCEDURE
  - INITIATE
  - GENERATE
  - TERMINATE

17.1

În lecţia 14 s-a descris modul în care se poate întocmi un raport utilizînd limbajul COBOL. Un program pentru listarea unui raport conţine, de regulă, un mare număr de operaţii privind controlul începutului şi sfîrşitului paginii, ca şi ale capitolelor şi subcapitolelor. Paginile trebuie să fie numerotate, se cer precauţii speciale la titluri şi antete, la totaluri generale şi parţiale, etc. Toate aceste operaţii fac ca în întocmirea programelor de listare să apară frecvent dificultăţi şi erori.

Pentru a uşura munca de întocmire a rapoartelor în COBOL, limbajul a fost prevăzut cu un capitol special — editorul de rapoarte.

17.2

Î. Urmăriţi raportul (pag. 247), produs în urma rulării programului „Program 17.1" (pag. 245).

1. Titlul raportului este . . . .
Raportul conţine . . . . pagini, Tranzacţiile sînt listate pe . . . . şi grupate pe . . . .

2. La sfîrşitul fiecărei . . . . şi a fiecărei . . . . . . se dă o valoare totală. Se mai dă, în coloana din dreapta, o valoare cumulată, iar la sfîrşitul raportului un . . . .

R. 1. DESFĂŞURĂTOR TRANZACŢII, două, dreapta jos, zile, luni
2. Zile, luni, total general.

17.3 Tranzacţiile din § 17.2 au structura pe care o putem urmări în programul sursă COBOL din programul 17.1 (vezi fişierul TRANZACŢII). Valoarea pentru o tranzacţie rezultă înmulţind conţinutul cîmpurilor CANTITATE şi PREŢ—UNITAR. Rezultatul va fi depus în cîmpul VALOARE din WORKING—STORAGE SECTION.

```
 COMPILE COBOL
STARTED
                                                    COBOL*ANS** 2, 6*

       IDENTIFICATION DIVISION.
       PROGRAM-ID. RWEXEMP.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT TRANZACTII     ASSIGN TO SYSIN.
           SELECT FISIER-RAPORT  ASSIGN TO SYSOUT.
       DATA DIVISION.
       FILE SECTION.
       FD  TRANZACTII LABEL RECORDS ARE OMITTED RECORDING F.
       01  TRANZACTIE.
        2  CAP          PIC XXXX.
        2  COD-ARTICOL  PIC X(8).
        2  TIP-TRANZ    PIC X.
        2  CANTITATE    PIC 9(5).
        2  PRET-UNITAR  PIC 9(4)V99.
        2  ZIUA         PIC 99.
        2  LUNA         PIC 99.
        2  ANUL         PIC 99.
       FD  FISIER-RAPORT LABEL RECORDS OMITTED RECORD 133
           REPORT IS LISTA-TRANZACTII.
       WORKING-STORAGE SECTION.
       77  VALOARE      PIC 9(9)V99 COMP-3.
       77  WSF          PIC 9 VALUE 0.
        88 SFIRSIT            VALUE 1.
       REPORT SECTION.
       RD  LISTA-TRANZACTII
              CONTROLS ARE FINAL LUNA ZIUA
              PAGE LIMIT 60 LINES
                 HEADING 1
                 FIRST DETAIL 8
                 LAST DETAIL 50
                 FOOTING 56.
       01  TYPE IS REPORT HEADING.
        2  LINE NUMBER 1 COLUMN 20 PIC X(23) VALUE
              'DESFASURATOR TRANZACTII'.
        2  LINE PLUS 1   COLUMN 20 PIC X(23) VALUE ALL '*'.
       01  TYPE IS PAGE HEADING.
        2  LINE PLUS 3.
         3  COLUMN 16 PIC X(11) VALUE 'COD ARTICOL'.
         3  COLUMN 29 PIC X(12) VALUE 'TIP  VALOARE'.
         3  COLUMN 48 PIC X(12) VALUE 'VAL.CUMULATA'.
       01  TYPE CONTROL HEADING LUNA.
        2  LINE PLUS 3.
         3  COLUMN 8  PIC X(5)   VALUE 'LUNA:'.
         3  COLUMN 13 PIC XX     SOURCE LUNA.
```

Programul 17.1.

COBOL*ANS** 2, 6*,

```
 2  LINE PLUS 1.
  3  COLUMN 8  PIC X(7)  VALUE ALL '-'.
01  TYPE CONTROL HEADING ZIUA.
 2  LINE PLUS 2.
  3  COLUMN 8  PIC X(5)  VALUE 'ZIUA:'.
  3  COLUMN 13 PIC XX    SOURCE ZIUA.
01  RIND-CURENT TYPE DETAIL LINE PLUS 1.
  3  COLUMN 17 PIC X(8)  SOURCE COD-ARTICOL.
  3  COLUMN 30 PIC X     SOURCE TIP-TRANZ.
  3  COLUMN 33 PIC Z(8)9.99 SOURCE VALOARE.
01  TYPE CONTROL FOOTING ZIUA.
 2  LINE PLUS 1.
  3  COLUMN 17 PIC X(11) VALUE 'TOTAL ZIUA'.
  3  COLUMN 28 PIC XX    SOURCE ZIUA.
  3 VAL-PE-ZI
     COLUMN 33 PIC Z(8)9.99 SUM VALOARE.
  3  COLUMN 48 PIC Z(8)9.99 SUM VALOARE RESET ON FINAL.
 2  LINE PLUS 1 COLUMN 8 PIC X(52) VALUE ALL '-'.
01  TYPE CONTROL FOOTING LUNA LINE PLUS 2.
  3  COLUMN 17 PIC X(11) VALUE 'TOTAL LUNA'.
  3  COLUMN 28 PIC XX    SOURCE LUNA.
  3  VAL-PE-LUNA
     COLUMN 33 PIC Z(8)9.99 SUM VAL-PE-ZI.
01  TYPE CONTROL FOOTING FINAL LINE PLUS 3.
  3  COLUMN 17 PIC X(15) VALUE 'TOTAL GENERAL :'.
  3  COLUMN 33 PIC Z(8)9.99 SUM VAL-PE-LUNA.
01  TYPE PAGE FOOTING LINE 60.
  3  COLUMN 53 PIC XXXX  VALUE 'PAG.'.
  3  COLUMN 57 PIC ZZ    SOURCE PAGE-COUNTER.
PROCEDURE DIVISION.
1   OPEN INPUT TRANZACTII OUTPUT FISIER-RAPORT.
    INITIATE LISTA-TRANZACTII
    PERFORM CITIRE
    PERFORM PRELUCRARE UNTIL SFIRSIT
    TERMINATE LISTA-TRANZACTII
    CLOSE TRANZACTII FISIER-RAPORT
    STOP RUN.
PRELUCRARE.
    MULTIPLY CANTITATE BY PRET-UNITAR GIVING VALOARE
    GENERATE RIND-CURENT
    PERFORM CITIRE.
CITIRE.
    READ TRANZACTII AT END MOVE 1 TO WSF.
```

Programul 17.1 (continuare)

```
RUN
STARTED

                 DESFASURATOR TRANZACTII
                 ***********************

               COD ARTICOL  TIP  VALOARE       VAL,CUMULATA

LUNA:04
-------

ZIUA: 1
            AB                A          1,00
            TOTAL ZIUA   1               1,00            1,00
--------------------------------------------------------------

ZIUA: 2
            AB1               B          4,00
            A                 C          6,00
            ABAC              A         30,00
            TOTAL ZIUA   2              40,00           41,00
--------------------------------------------------------------

ZIUA: 5
            ABA               B         20,00
            AB2               C         30,00
            TOTAL ZIUA   5              50,00           91,00
--------------------------------------------------------------

ZIUA:10
            AC                A        200,00
            AB                B        400,00
            TOTAL ZIUA  10             600,00          691,00
--------------------------------------------------------------

            TOTAL LUNA 04              691,00

LUNA:05
-------

ZIUA: 2
            A                 C        900,00
            TOTAL ZIUA   2             900,00         1591,00
--------------------------------------------------------------

ZIUA: 3
            ACAB              A       1200,00
            TOTAL ZIUA   3            1200,00         2791,00
--------------------------------------------------------------
```

Programul 17.1 (continuare)

```
COD ARTICOL  TIP  VALOARE        VAL.CUMULATA


ZIUA: 4
          AAA          B       5000,00
          A            C        400,00
          A2           A        900,00
          TOTAL ZIUA   4       6300,00          9091,00
---------------------------------------------------------

ZIUA:10
          A3           B       1600,00
          TOTAL ZIUA  10       1600,00         10691,00
---------------------------------------------------------

          TOTAL LUNA 05       10000,00


          TOTAL GENERAL :     10691,00
```

Programul 17.1 (continuare)

Un *raport* se descrie, conform convențiilor generatorului de rapoarte COBOL, prin două entități:

— *un fișier*, declarat în FILE SECTION, cu FD, și avînd între altele clauza REPORT urmată de numele raportului;

— *raportul*, precedat de RD, în cadrul secțiunii REPORT SECTION, urmat de clauze speciale și descrierea rîndurilor.

17.5 *Î.* 1. În programul 17.1 numele fișierului asociat raportului este . . . . iar numele raportului . . . .

2. În FILE—CONTROL se declară printr-o frază SELECT doar . . . .

R. 1. FIȘIER—RAPORT, LISTA—TRANZACȚII
2. Fișierul asociat: FIȘIER—RAPORT

17.6 Rubrica de descriere a fișierului asociat raportului are formatul general:

FD nume—fișier
 [clauză **BLOCK** CONTAINS]
 [clauză **RECORD** CONTAINS]
 clauză **LABEL RECORDS**
 clauză REPORT . . . .

Clauzele BLOCK, RECORD și LABEL sînt aceleași cu cele discutate la rubrica de descriere a fișierelor.

Clauza RECORD stabilește lungimea rîndului. Prin absență această lungime se ia de 133 caractere.

Clauza REPORT are formatul:

{**REPORT** IS | **REPORTS** ARE} nume—raport...

și este obligatoriu. Numele de raport este cel asociat unei rubrici RD în secțiunea REPORT.

17.7 Rubrica de descriere a raportului are formatul **RD** nume—raport
[clauza **CODE**]
[clauza **CONTROL**]
[clauza **PAGE LIMIT**].

Clauza CODE se folosește în cazul în care se dorește scrierea mai multor rapoarte avînd același fișier asociat și nu se va detalia aici. În caz de omisiune se consideră că fișierul se asociază unui singur raport.

Clauzele CONTROL și PAGE LIMIT necesită o discuție mai detaliată care se va face în continuare.

17.8 O noțiune importantă în cadrul editorului de rapoarte COBOL este cea de *control*; termenul este utilizat aici într-un sens diferit de cel uzual. În programul 17.1 controalele sînt ,,luna'' și ,,ziua''. În legătură cu controalele putem remarca următoarele:

1. Controalele sînt obligatoriu într-o *ierarhie*; în cazul nostru ,,luna'' este superior ierarhic lui ,,ziua''.

2. Datele care servesc producerii raportului, în cazul nostru tranzacțiile, trebuie să fie sortate sau măcar grupate în funcție de controale.

3. Ori de cîte ori controlul se modifică are loc o situație numită *schimbare* sau *ruptură de control* (de pildă schimbarea zilei sau a lunii de la o tranzacție la următoarea). În cazul unei rupturi de control se cere de regulă efectuarea unor operații suplimentare pe lîngă tipărirea obișnuită a rîndului (de pildă totaluri, salt la pagină nouă, etc.).

4. Un control ierarhic inferior (de ex. ,,ziua'') se poate schimba de mai multe ori înainte de schimbarea unui control ierarhic superior (de ex. ,,luna'') dar la schimbarea unui control superior se consideră automat că a avut loc o schimbare a tuturor controalelor ierarhic inferioare.

17.9 *Î.* În programul 17.1 există . . . . controale numite (în ordine ierarhică) . . . .

R. două, luna, ziua.

17.10 În rapoarte întîlnim următoarea structură, asociată unui anumit control dat:

— *antet*, pentru o valoare a acelui control;
— *rînduri*, pentru toate datele care au aceeași valoare a controlului;
— *sfîrșit*, pentru respectiva valoare a controlului.

La o ruptură de control se va tipări sfîrșitul pentru valoarea precedentă a controlului și începutul pentru noua valoare.

17.11 *Î.* 1. Examinați raportul produs în cadrul programului 17.1. Antetul controlului ,,luna'' conține . . . .
iar sfîrșitul aceluiași control . . . .
2. Antetul controlului ,,ziua'' conține . . . .
iar sfîrșitul . . . .

R. 1. LUNA: numărul lunii şi o subliniere scurtă cu liniuţe, TOTAL LUNA numărul lunii şi o valoare totală.
2. ZIUA: numărul zilei, TOTAL ZIUA numărul zilei şi o subliniere lungă cu liniuţe.

17.12

Deci antetul sau sfîrşitul unor controale conţine unul sau mai multe rînduri: în unele cazuri el ar putea lipsi complet. Se defineşte în acest sens noţiunea de *grup de editare*, care poate fi un rînd, o succesiune de rînduri sau un „grup nul", care nu se imprimă. Grupurile de editare pot fi de următoarele tipuri (tabelul 17.1):
— grup antet;
— grup sfîrşit;
— grup detaliu.

Tabelul 17.1

Tipul grupului	Denumire completă	Den. prescurtată
antet de raport	REPORT HEADING	RH
antet de pagină	PAGE HEADING	PH
antet de control	CONTROL HEADING	CH
detaliu (rînduri curente)	DETAIL	DE
sfîrşit de control	CONTROL FOOTING	CF
sfîrşit de pagină	PAGE FOOTING	PF
sfîrşit de raport	REPORT FOOTING	RF

Un *grup detaliu* conţine rîndul curent sau un grup de rînduri curente care se tipăresc împreună.

17.13 *Grupurile antet* şi *sfîrşit* nu se definesc doar pentru controale; pot exista antet şi sfîrşit de *pagină* sau un antet şi un sfîrşit pentru *întregul raport*.

Întrebările de mai jos se referă la raportul „DESFĂŞURĂTOR TRANZACŢII" din programul 17.1.

Î. 1. Care este grupul antet de raport?
2. Care este grupul antet de pagină?

R. 1. 2 rînduri: „DESFĂŞURĂTOR TRANZACŢII" şi o subliniere din asteriscuri
2. „COD ARTICOL TIP VALOARE VAL. CUMULATA"

17.14 Un raport (descris printr-o rubrică RD) constă dintr-unul sau mai multe grupuri de editare. Fiecare grup de editare se descrie printr-o ierarhie de rubrici, într-un mod asemănător cu cel utilizat la descrierea înregistrărilor unui fişier.

Propunem cititorului să folosească totdeauna următoarea regulă de atribuire a numerelor de nivel:

01 — pentru grupurile de editare

02 — pentru rînduri în cadrul grupurilor de editare, dacă grupul conține mai multe rînduri

03—49 — pentru componente (ierarhizate) din cadrul unui rînd.

Un grup de editare, un rînd sau un element pot avea un nume. În cazul în care nu este necesar un nume, el va fi omis.

Grupurile de editare au, în mod obligatoriu, în descrierea lor clauza TYPE, iar rîndurile clauza LINE.

17.15 Î. În programul RWEXEMP (programul 17.1) să se identifice grupele de editare, rîndurile și elementele cerute mai jos:

1. Între ce rînduri ale programului sursă COBOL se află descrise grupele de editare și componentele lor?
2. Cîte grupuri de editare sînt definite?
3. Indicați numele unui grup de editare
4. Numărați cîte rînduri conține fiecare grup de editare, și scrieți aceste numere în ordinea definirii grupelor de editare
5. Indicați numărul de ordine al acelor grupuri de editare (avînd un singur rînd) la care rîndul este definit în cadrul aceleiași rubrici cu grupul de editare. Cu ce număr de nivel sînt definite rîndurile în această situație?

R. 1. 34 la 75
2. 9
3. RÎND—CURENT, celelalte grupuri de editare nu au nume
4. 2,1,2,1,1,2,1,1,1,
5. 5,7,8,9. La aceste grupuri de editare nu am utilizat nivelul 02, clauzele TYPE și LINE apărînd ambele la rubricile cu numărul de nivel 01.

17.16 Clauza TYPE care descrie o grupă de editare are următorul format:

```
          ⎧ ⎧REPORT HEADING ⎫                          ⎫
          ⎪ ⎩RH             ⎭                          ⎪
          ⎪ ⎧PAGE HEADING   ⎫                          ⎪
          ⎪ ⎩PH             ⎭                          ⎪
          ⎪ ⎧CONTROL HEADING⎫   ⎧identificator⎫        ⎪
          ⎪ ⎩CH             ⎭   ⎩FINAL        ⎭        ⎪
TYPE IS   ⎨ ⎧DETAIL         ⎫                          ⎬
          ⎪ ⎩DE             ⎭                          ⎪
          ⎪ ⎧CONTROL FOOTING⎫   ⎧identificator⎫        ⎪
          ⎪ ⎩CF             ⎭   ⎩FINAL        ⎭        ⎪
          ⎪ ⎧PAGE FOOTING   ⎫                          ⎪
          ⎪ ⎩PF             ⎭                          ⎪
          ⎪ ⎧REPORT FOOTING ⎫                          ⎪
          ⎩ ⎩RF             ⎭                          ⎭
```

Clauza, pe lîngă faptul că defineşte un grup de editare, indică tipul acestuia şi condiţiile în care urmează să fie tipărit.

Tipurile de grupuri de editare sînt date în tabelul 17.1.

După cum se vede din formatul clauzei TYPE, se poate folosi în definiţie atît denumirea completă a tipului de grup cît şi denumirea prescurtată.

Remarcă

Deoarece într-un raport putem avea mai multe controale la grupurile CH şi CF trebuie indicat numele controlului (prin „identificator"). Pe lîngă controalele definite prin date, se poate specifica şi un control special numit FINAL care se consideră că are o ruptură de control înainte de prelucrarea primei înregistrări şi după prelucrarea ultimei înregistrări.

17.17 *Î.* Indicaţi numărul de ordine din programul RWEXEMP al grupului de editare de tipul:
1. Antet şi sfîrşit de pagină
2. Antet şi sfîrşit de raport
3. Rînd curent (detaliu)

R. 1. 2,9
2. 1, nu există
3. 5

17.18 Clauza CONTROL din rubrica RD (vezi şi 17.7) are formatul:

```
{CONTROL IS  }  {FINAL                                            }
{CONTROLS ARE}  {identificator—1 [identificator—2] ...            }
                {FINAL identificator—1 [identificator—2] ...      }
```

Identificatorii reprezintă numele controalelor utilizate în raport, în ordine ierarhică descrescătoare de la stînga la dreapta. Controalele trebuie să fie nume de date definite în secţiunile FILE sau WORKING—STORAGE.

17.19 *Î.* 1. Cîte controale se folosesc în programul RWEXEMP şi care este numele lor?
2. Au toate controalele grup de editare tip antet şi sfîrşit?

R. 1. trei: FINAL, LUNA, ZIUA, în această ierarhie
2. Controlul FINAL nu are decît CF, celelalte au ambele tipuri de grupuri de editare.

17.20 Clauza PAGE LIMIT descrie restricţiile stabilite privind dimensiunile unei pagini din raport, stabilind zonele în care se vor tipări diferitele grupuri de antet, sfîrşit şi detaliu.

Formatul este:

```
PAGE {LIMIT IS  }  întreg—1   {LINE }
     {LIMITS ARE}             {LINES}
[HEADING întreg—2]
[FIRST DETAIL întreg—3]
[LAST DETAIL întreg—4]
[FOOTING întreg—5]
```

O pagină de imprimantă are de obicei 72 sau 96 de rînduri (după densitatea rîndurilor la care este reglată imprimanta). Să ne imaginăm că am numerotat rîndurile unei pagini de la 1 la un număr maxim prevăzut, ţinînd cont că linia nr. 1 va fi cea la care sare imprimanta la o comandă „salt la pagină nouă". În acest caz semnificaţia întregilor din formatul clauzei PAGE este următoarea:

— **întreg—2**, numărul primei linii pe care se poate tipări un rînd dintr-un grup de editare tip antet de pagină;

— **întreg—3**, numărul primei linii pe care se poate tipări un rînd dintr-un grup de editare tip detaliu sau sfîrşit;

— **întreg—4**, numărul ultimei linii pe care se (mai) poate tipări un rînd, dintr-un grup de editare tip antet de control sau detaliu;

— **întreg—5**, numărul ultimei linii pe care se poate tipări un rînd dintr-un grup de editare tip sfîrşit control;

— **întreg—1**, numărul ultimei linii pe care se poate tipări un rînd într-o pagină.

În caz de omisiune **întreg—2** se ia egal cu 1; dacă nu se specifică **întreg—3** el se ia egal cu **întreg—2**; dacă unul dintre **întreg—4** sau **întreg—5** nu este dat, el va fi luat egal cu celălalt; dacă nici unul nu este specificat, amîndoi se iau egali cu **întreg—1**.

Trebuie să fie îndeplinită, în mod obligatoriu inegalitatea:

1 ⩽ întreg—2 ⩽ întreg—3 ⩽ întreg—4 ⩽ întreg—5 ⩽ întreg—1 ⩽ 999

17.21 *Î.* **Precizaţi pentru raportul produs de programul RWEXEMP (programul 17.1) următoarele date:**
1. **Cîte rînduri se pot tipări cel mult pe o pagină?**
2. **Între ce rînduri se poate tipări rîndul curent?**
3. **Care-i ultimul rînd pe care se poate tipări „totalul pe o lună"?**

R. 1. 60
2. 8 şi 50
3. 56, deoarece „totalul pe o lună" va figura într-un grup de editare tip sfîrşit control.

17.22 Rubrica de descriere a unui grup de editare are următorul format:

01 [nume dată] clauza **TYPE**
[clauza **NEXT GROUP**]
[clauza **LINE**]
[clauza **USAGE**]

Numele de dată este necesar doar dacă grupul de editare este apelat în mod explicit printr-un GENERATE, USE sau un alt mod, în cadrul programului.

Clauza USAGE este cea de la rubrica de descriere a datelor. În acest caz nu poate fi decît DISPLAY.

17.23 Clauza NEXT GROUP indică condiţiile de spaţiere după tipărirea liniilor care compun grupul.

Formatul clauzei este:

NEXT GROUP IS { întreg—1 | **PLUS** întreg—2 | ON **NEXT PAGE** }

indicînd că următorul grup trebuie tipărit într-unul dintre următoarele moduri:

- a, începînd cu linia întreg—1 din pagină;
- b, cu întreg—2 rînduri după terminarea grupului de editare curent;
- c, la începutul paginii următoare (NEXT PAGE).

Întregii trebuie să fie pozitivi și diferiți de zero. Opțiunea ON NEXT PAGE nu este permisă în cazul grupurilor de editare tip antet sau sfîrșit de pagină și nici în cazul sfîrșitului de raport.

17.24 *Clauza LINE* definește un rînd. Ea poate apare în aceeași rubrică cu TYPE doar dacă grupul de editare conține un singur rînd. Se poate scrie și la o rubrică de alt număr la nivel decît 01.

Formatul clauzei LINE este

LINE NUMBER IS { întreg—1 [ON **NEXT PAGE**] | **PLUS** întreg—2 | ON **NEXT PAGE** }

Se cere ca întreg—1, întreg—2 să fie strict pozitive și mai mici decît 1000. ON NEXT PAGE implică saltul la pagină nouă. Această opțiune nu este permisă decît pentru primul rînd dintr-un grup.

Prima opțiune specifică pe care anume linie din pagină se va tipări rîndul. Acest mod de definire poartă numele de *spațiere absolută*. A doua opțiune permite o *spațiere relativă* adică întreg—2 indică numărul de linii cu care se dorește să avanseze imprimanta, înainte de a se tipări rîndul la care se referă clauza LINE, față de ultimul rînd deja tipărit. A treia opțiune indică un salt la primul rînd de pe pagina următoare.

Opțiunea NEXT PAGE nu se permite în cazul unui rînd dintr-un grup antet de raport sau dintr-un grup antet sau sfîrșit de pagină.

17.25 Un antet de raport sau de pagină poate conține o spațiere absolută (implicînd de regulă un salt la pagina următoare) sau una relativă. În cazul al doilea spațierea este relativă față de „întreg—2" definit prin clauza PAGE LIMITS din rubrica RD, sau în cazul antetului de pagină, față de ultimul rînd din antetul de raport, dacă s-a tipărit pe aceeași pagină.

17.26 *Î.* Se urmărește modul în care a fost definit raportul din programul RWEXEMP:

1. Cîte rînduri sînt definite cu spațiere absolută și cîte cu spațiere relativă?
2. Față de cine este relativă poziția rîndului din antetul de pagină? Cu cîte linii?

R. 1. 2 rînduri cu spațiere absolută (primul rînd din antetul de raport și rîndul de sfîrșit pagină) și celelalte 10 cu spațiere relativă.
2. Pe prima pagină față de sublinierea titlului iar pe celelalte față de rîndul 1, cu un avans de trei linii.

17.27 Editorul de rapoarte COBOL poate genera două contoare speciale care nu sînt definite explicit în diviziunea DATA, şi anume:

— LINE—COUNTER destinat numărării liniei la care s-a ajuns cu imprimarea în cadrul paginii;

— PAGE—COUNTER destinat numărării paginilor de imprimantă.

Cele două contoare pot fi folosite pentru operaţii în diviziunea de prelucrare. Dimensiunile contoarelor se stabilesc de către compilator în funcţie de utilizările lor.

Contorul LINE—COUNTER este mărit automat pe măsură ce se generează liniile la imprimantă şi readus la 1 ori de cîte ori se face salt la pagină nouă.

PAGE—COUNTER conţine iniţial 1 şi la fiecare salt la pagină nouă se măreşte cu 1.

17.28 Rubrica de descriere a unei componente dintr-o grupă de editare are formatul:

```
nr—nivel        [nume dată]
                [clauza BLANK WHEN ZERO]
                [clauza COLUMN]
                [clauza GROUP]
                [clauza JUSTIFIED]
                [clauza LINE]
                [clauza PICTURE]
                [clauza RESET]
                [         { SOURCE } ]
                [ clauza  { SUM    } ]
                [         { VALUE  } ]
                [USAGE DISPLAY]
```

unde:

numărul de nivel poate fi şi 1 dar îl recomandăm între 2 şi 49.

Clauzele BLANK WHEN ZERO, JUSTIFIED, PICTURE şi USAGE DISPLAY sînt aceleaşi cu cele de la rubricile de descriere a datelor.

Numele componentei trebuie specificat doar dacă se face o referire expresă la ea dintr-o altă parte a programului.

17.29 *Î.* Indicaţi numele unor componente care nu sînt rînduri, în descrierea raportului din programul RWEXEMP.

R. VAL—PE—ZI, VAL—PE—LUNA sînt singurele nume de componente avînd caracteristicile cerute.

17.30 *Clauza COLUMN* are formatul:

COLUMN NUMBER IS întreg

Întregul trebuie să fie strict pozitiv şi max. 132. Se aplică componentelor elementare din rînd. Întregul indică poziţia în rînd a celui mai din stînga caracter al componentei. Dacă clauza nu se specifică, respectiva componentă nu va fi tipărită. Componentele astfel definite nu este permis să se suprapună. Zonele libere dintre componente se vor umple cu blancuri.

17.31 *Clauza GROUP INDICATE* are formatul:
GROUP INDICATE

Se utilizează la o componentă în cadrul unui grup detaliu. Componenta va fi tipărită la imprimantă numai la prima generare a grupului detaliu după o ruptură de pagină sau control sau de la începutul raportului, la celelalte generări locul componentei fiind ocupat de spații.

17.32 Clauzele VALUE, SOURCE și SUM se aplică componentelor elementare. Ele sînt reciproc exclusive dar una dintre ele este obligatorie dacă se dorește imprimarea componentei.

Componentele elementare sînt deci de trei tipuri: VALUE, SOURCE sau SUM.

Clauza VALUE are formatul:
VALUE IS literal

Componenta se tipărește cu valoarea literalului ori de cîte ori se generează grupul care conține componenta. Excepție face cazul în care se asociază un GROUP GENERATE; în acest caz se poate imprima această valoare sau spațiu.

17.33 Clauzele VALUE și PICTURE trebuie să concorde întocmai ca în cazul rubricilor de descriere a datelor.

17.34 *Clauza SOURCE* are formatul:

$$\textbf{SOURCE} \text{ IS } \begin{Bmatrix} \text{identificator} \\ \textbf{TALLY} \end{Bmatrix}$$

reprezintă un MOVE implicit de la identificator, sau registrul TALLY, la componenta în descrierea căreia intră clauza SOURCE. Mutarea se face imediat înainte de tipărire.

Exemplu
3 COLUMN 33 PIC Z(8)9.99 SOURCE VALOARE.

Cititorul este invitat să consulte programul RWEXEMP (program 17.1). Acolo componenta de mai sus face parte din RIND-CURENT. Din coloana 33 se va tipări conținutul cîmpului VALOARE editat corespunzător formatului dat prin PIC.

17. 5 *Clauza SUM* are formatul:

$$\textbf{SUM} \begin{Bmatrix} \text{identificator-1} \\ \textbf{TALLY} \end{Bmatrix} \left[\begin{Bmatrix} \text{identificator-2} \\ \textbf{TALLY} \end{Bmatrix} \right] \ldots \quad [\textbf{UPON} \text{ nume-dată} \ldots] \ldots$$

Definește pe de o parte un contor de însumare la nivelul unei componente elementare în cadrul unui grup tip sfîrșit, pe de altă parte componenta care urmează să se tipărească. Ori de cîte ori se va efectua tipărirea, conținutul din momentul respectiv al contorului se mută în componenta de tipărit, eventual cu editare.

Exemplu:
3 VAL-PE-ZI
COLUMN 33 PIC Z(8)9.99 SUM VALOARE.

În acest caz VAL-PE-ZI este numele contorului de însumare în care se acumulează conținutul cîmpului VALOARE (din secțiunea WORKING-STORAGE în RWEXEMP).

Componenta de tipărit nu are nume, se situează în poziția 33 a primului rînd din grupul sfîrșit de control „ziua" și editează valoarea controlului conform șablonului din PIC.

Un identificator specificat într-o clauză SUM trebuie să apară exact în aceeași formă (inclusiv calificare, indici) într-una dintre următoarele situații:

a) într-o clauză SOURCE într-un grup detaliu;
b) să fie contor de sumare într-un grup sfîrșit de control de nivel inferior;
c) să fie contor de sumare în același grup sfîrșit de control.

17.36 *Î.* Unde sînt specificați identificatorii din clauzele SUM în programul RWEXEMP?

R. Există 4 clauze SUM în program, folosind următorii 3 identificatori:
— VALOARE — care apare în clauza SOURCE într-o componentă din RIND-CURENT
— VAL-PE-ZI și VAL-PE-LUNĂ care sînt contoare de sumare definite în grupuri de control de nivel inferior.

17.37 Principalele reguli după care are loc cumularea datelor în contoare sînt următoarele:

— un contor în al cărui SUM există un identificator SOURCE într-un grup detaliu, se va mări cu valoarea corespunzătoare identificatorului ori de cîte ori se generează respectivul grup detaliu. Chiar dacă în grupul detaliu identificatorul apare în mai multe clauze SOURCE el va fi adunat doar o singură dată.

— un contor avînd în SUM un contor definit într-un grup sfîrșit de control de nivel inferior se mărește la fiecare tipărire a acestui grup sfîrșit de control.

— în cazul unor contoare aparținînd aceluiași grup de control adunările se fac înainte de tipărire, în ordinea în care contoarele sînt definite în cadrul grupei.

17.38 Aducerea contoarelor la zero se face automat atît la începutul programului cît și după fiecare tipărire a contorului respectiv, în afară de cazul că se specifică altfel printr-o clauză RESET.

Format:

RESET ON { identificator | **FINAL** }

Identificatorul (ca și FINAL) reprezintă un control.

Clauza se folosește în asociație cu un contor definit prin SUM: ea determină ca contorul să fie readus la zero nu la o ruptură a controlului asociat grupei căreia îi aparțin clauzele SUM și RESET ci la ruptura controlului indicat în clauza RESET. Acest control trebuie să fie superior (în ierarhia controalelor) celei asociate grupei în care s-a definit SUM și RESET.

17.39 *Î.* Care sînt asemănările și deosebirile dintre cele două contoare definite în grupul sfîrșit de control „ZIUA", în programul RWEXEMP?

R. Cele două contoare sînt: VAL-PE-ZI și un contor fără nume (COLUMN 48). Ambele acumulează aceeași valoare și o vor tipări după o editare prin același format. Primul contor se reduce însă la zero la fiecare schimbare a controlului ZIUA pe cînd al doilea crește neîncetat pînă la sfîrșitul programului (FINAL).

17.40 În diviziunea PROCEDURE editorul de rapoarte permite utilizarea verbelor:
- INITIATE
- GENERATE
- TERMINATE

17.41 *Verbul INITIATE* are formatul:

INITIATE nume raport-1 [nume raport-2] ...

pune toate contoarele definite cu SUM și registrul LINE-COUNTER pe zero, iar PAGE-COUNTER (dacă există) pe 1. Nu este permis să se execute un alt INITIATE pentru un raport înainte de un TERMINATE pentru același raport.

17.42 *Verbul TERMINATE* are formatul:

TERMINATE nume raport-1 [nume raport-2] ...

și generează toate grupurile de sfîrșit control în ordinea crescătoare a ierarhiei controalelor, ca și grupurile sfîrșit pagină și sfîrșit raport. Nu este permis să se execute de două sau mai multe ori consecutiv.

Remarcă

INITIATE și TERMINATE presupun și OPEN și CLOSE pentru fișierul raport asociat.

17.43 *Verbul GENERATE* are formatul:

GENERATE { nume-grup-detaliu | nume-raport }

Cele două opțiuni duc la două tipuri de rapoarte
- raport detaliat;
- raport rezumat.

În raportul rezumat se efectuează toate operațiile specifice raportului detaliat, cu excepția tipăririi rîndurilor din grupurile detaliu.

Operațiile care se produc cu ocazia unei GENERATE sînt următoarele:

— se mărește și se testează LINE-COUNTER și/sau PAGE-COUNTER pentru a produce grupul de editare sfîrșit de pagină sau antet de pagină după tipărirea rîndului;

— se testează controalele pentru a produce, dacă e cazul grupele de sfîrșit și antet de control necesare;

— se acumulează în contoarele definite prin SUM valorile identificatorilor specificați (aceleași contoare vor fi readuse la zero după tipărire în cazul rupturii controlului asociat fără clauza RESET);

— se tipăresc grupele de editare necesare; la primul GENERATE se tipăresc:

- grupul antet de raport;
- grupul antet de pagină;
- toate grupurile antet de control de la FINAL, controlul major pînă la controlul minor în ierarhie;
- grupul detaliu citat.

La un GENERATE care constată o ruptură de control se tipăresc:

— toate grupele sfîrşit de control de la cel minor, crescător, pînă la cel asociat rupturii;

— toate grupele antet de control de la cel asociat rupturii, descrescător, pînă la controlul minor;

— grupul detaliu citat în GENERATE.

În caz că nu există rupturi de control se generează doar grupul de control detaliu.

17.44 *Î.* Întrebările următoare se referă la programul RWEXEMP:

1. Raportul este detaliat sau rezumat?
2. Ce grupe de editare se generează la prima execuţie a verbului GENERATE?
3. Ce grupe de editare se generează la schimbarea controlului LUNA?
4. Ce grupe de editare se generează la executarea verbului TERMINATE?

R. 1. Detaliat, deoarece în GENERATE se specifică numele grupului detaliu.
2. Antet raport, antet pagină, antet luna, antet ziua şi primul rînd detaliu.
3. Sfîrşit ziua, sfîrşit luna, antet luna, antet ziua şi rîndul detaliu.
4. Sfîrşit ziua, sfîrşit luna, sfîrşit FINAL şi sfîrşit pagină.

Cititorul este rugat să verifice cele de mai sus pe raportul rezultat din rularea programului 17.1.

17.45 În editarea rapoartelor se pot folosi declarative. Un declarativ reprezintă unul sau mai multe paragrafe care conţin operaţii care se execută înaintea generării unui grup de editare.

Format:

nume secţiune **SECTION USE BEFORE REPORTING** nume dată.

```
•
•} operaţii în cadrul declarativului
•
```

Numele de dată este cel al grupului de editare înaintea generării căruia se execută operaţiile din cadrul declarativului. Acest grup poate fi oricare în afara unui grup detaliu.

Nu este permisă referirea în cadrul unui declarativ la proceduri din afara declarativului, nici utilizarea verbelor INITIATE, TERMINATE, GENERATE.

Se poate suprima imprimarea grupei de editare asociate incluzînd în cadrul declarativului unul dintre enunţurile:

MOVE 1 TO PRINT-SWITCH sau
SUPPRESS PRINTING

LECŢIA a 18-a LIMBAJUL DE COMANDĂ ŞI CARTELELE CU PUNCT

- generalităţi
- convenţii de descriere a cartelelor cu punct
- ordonarea cartelelor de comandă:
  cazuri simplificate
- cartele de comandă: JOB, COMPILE, LINK, RUN, EOJ, EOF, LABEL, INIT, ASSIGN, ALLOC, DELETE, FETCH

18.1 Execuţia unui program comportă trecerea acestuia prin mai multe faze, pentru care pot fi specificate anumite opţiuni dorite de utilizator. În faza finală de pregătire a execuţiei se realizează asocierea la fişierele descrise şi utilizate în program a unităţilor periferice şi a fişierelor fizice de pe acestea după care urmează execuţia propriu-zisă a programului.

Particularităţile unei execuţii sînt transmise sistemului de exploatare prin intermediul cartelelor de comandă*. Acestea pot fi prelucrate de monitorul de înlănţuiri şi de editorul de legături (cartele cu punct '.' în coloana 1) sau de programul bibliotecar şi alte programe utilitare ale sistemului (cartele cu caracterul '%' în prima coloană).

Cartelele de comandă sînt citite din fişierul de lucrări ale sistemului verificate sintactic şi interpretate. Apariţia unei erori de orice natură într-o cartelă de comandă produce abandonarea cartelei, a fazei şi a lucrării.

18.2 *Î.* Prin intermediul cartelelor de comandă programatorul specifică sistemului . . . . referitoare la rularea unui program şi descrie anumite informaţii referitoare la . . . . . . prelucrate în program.

R. — fazele de execuţie şi opţiuni în cadrul lor;
— fişierele.

18.3 *Î.* Cartelele de comandă se găsesc în . . . . şi pot conţine în prima coloană unul din caracterele . . . .

R. — fişierul de lucrări ale sistemului;
— punct ('.') sau procent ('%').

Limbajul de comandă îşi are descrise cartelele utilizînd convenţii de descriere a acestora (metalimbaj). Principalele convenţii sînt următoarele:

— cuvintele trebuie scrise cu majuscule aşa cum apar în format;

— parantezele drepte indică prezenţa facultativă a argumentului pe care îl conţin;

---

* Care se găsesc în fişierul de lucrări al sistemului de operare.

— acoladele cuprind două sau mai multe variante (alternative) aranjate pe verticală dintre care una este obligatorie de ales;

— elementul subliniat indică opţiunea implicită pe care o are sistemul cînd argumentul respectiv este absent.

Celelalte convenţii sînt similare celor de la limbajul COBOL

18.4 *Î.* Argumentul sau argumentele încadrate între paranteze drepte sînt . . . ., iar pentru cele între acolade . . . . de specificat.

R. — opţionale de utilizat;
— o variantă este obligatorie.

18.5 *Î.* Argumentul sau argumentele urmate de grupul de trei puncte . . . ., iar cele subliniate reprezintă . . . .

R. — pot părea de un număr oarecare de ori în format;
— opţiunea implicită, pe care o ia sistemul în lipsa acelui argument.

18.6 Cartelele de comandă cu punct au următorul format general:

```
. [Eticheta] Ident Argumentul1, Argumentul2 ... [Comentarii]
↑            ↑
Col. 1       Col.11
```

De exemplu:

```
.       JOB PROGRAM1,AN :CCO1,PN :ANDRA
.       COMPILE COBOL
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.       EOJ
```

unde:

— în coloana 1, caracterul punct ('.') care reprezintă însemnul cartelelor de comandă ale sistemului de exploatare;

— în coloanele 2 la 9 facultativ se poate trece o etichetă, ce va fi aliniată la stînga;

— în coloana 10 obligatoriu spaţiu;

— din coloana 11 se trece identificatorul cartelei de comandă, care indică tipul operaţiei ce trebuie executată şi care este urmat obligatoriu de un spaţiu;

— urmează argumentele cartelei de comandă care sînt diferite ca număr în funcţie de tipul cartelei. Nu se admite nici un caracter spaţiu în zona argumentelor sau a parametrilor în afară de cazul în care spaţiul se află într-un şir de caractere definit prin lungimea sa sau încadrat între apostrofuri.

Argumentele sînt separate prin virgule şi nu pot depăşi coloana 71 a cartelei, continuarea pe următoarea cartelă indicîndu-se printr-un asterisc în coloana 72 a cartelei ce trebuie continuată. Cartela de continuare conţine punct de coloana 1 iar specificarea argumentelor începe din coloana 20. Continuarea unei cartele nu se face prin separarea unui argument pe două cartele, în acest caz este admisă înserarea unor spaţii între virgula ce urmează ultimului argument de pe cartelă şi asteriscul din coloana 72. O cartelă de continuare nu mai conţine etichetă şi identificator.

18.7 *Î.* Cartelele de comandă care au continuare conțin . . . . iar cartela de continuare nu conține . . . . și argumentele ei sînt specificate din . . . .

R. — caracterul asterisc ('*') în coloana 72;
— eticheta și identificator;
— coloana 20.

18.8 *Î.* Cartelele de comandă cu punct au specificat începînd din coloana 11 . . . . care este urmat de un caracter spațiu, după care urmează argumentele cartelei ce sînt separate prin . . . iar acestea nu pot depăși coloana . . a cartelei

R. — identificatorul cartelei de comandă;
— virgule;
— 71.

18.9 După funcțiile pe care le îndeplinesc, cartelele de comandă pot fi grupate în trei categorii principale:

— Cartele de comandă referitoare la exploatarea lucrărilor JOB, RUN, EOJ, FETCH și altele;

— Cartele de comandă referitoare la punerea în lucru a programelor de traducere și a editorului de legături COMPILE, LINK și altele;

— Cartele de comandă referitoare la prelucrarea fișierelor INIT, ALLOC, ASSIGN, LABEL, DELETE și altele.

Observație: în continuare se va prezenta modul de utilizare pentru aceste cartele și parametri mai importanți iar pentru descrierea completă a cartelelor de comandă și a parametrilor respectivi consultați Monitorul Sistemului SIRIS—3.

18.10 *Î.* În general cartelele de comandă se pot grupa . . . . după cum se referă la . . . .

R. — în trei categorii principale;
— exploatarea lucrărilor, la punerea în lucru a programelor de traducere și a editorului de legături sau la prelucrarea fișierelor.

18.11 Etapele execuției unui program (compilarea, editarea de legături, execuția) sînt declanșate în urma citirii din fișierul de lucrări al sistemului a anumitor cartele de comandă specifice pentru fiecare etapă.

18.12 *Î.* Etapa de compilare a unui program se poate realiza prin specificarea unor cartele de comandă care sînt asociate modulului sursă. Care sînt acestea și cum este succesiunea lor?

R. Pentru compilarea unui program sursă sînt necesare următoarele cartele de comandă:

```
• JOB ...
• COMPILE COBOL
    {Cartelele programului sursă COBOL}
• EOJ
```

Dacă în urma compilării apar erori a căror gravitate împiedică trecerea la execuţia următoarei etape (editarea de legături), urmează corectarea acestora şi se reia compilarea pînă la eliminarea lor.

18.13 Editarea legăturilor este determinată de citirea cartelei de comandă LINK. După executarea editării de legături programul este pregătit pentru execuţia propriu-zisă care este declanşată de citirea cartelei RUN.

18.14 De exemplu pentru execuţia unui program în care datele de intrare de pe cartele sînt citite prin ACCEPT iar rezultatele prelucrării acestora sînt afişate la imprimantă cu DISPLAY sînt necesare următoarele cartele de comandă cu punct :

```
.        JOB
.        COMPILE COBOL
            {modul sursă COBOL
.        LINK
.        RUN NL : 3000,TIME : 15
            {cartele cu datele de intrare
.        EOJ
```

Observaţie : În acest program, deoarece datele prelucrate nu sînt considerate ca aparţinînd unor fişiere nu apar cartele de comandă referitoare la fişiere. În cartela RUN argumentul NL : 3000 semnifică imprimarea a maxim 3000 de rînduri iar TIME : 15 că execuţia programului poate dura cel mult 15 minute.

18.15 Între cartelele LINK şi RUN se pot introduce cartelele de comandă referitoare la fişierele prelucrate de program. (De exemplu INIT, DELETE, ALLOC, ASSIGN, LABEL).

Dacă programul COBOL are de prelucrat un fişier pe cartele atunci cartelele de date urmează cartelei RUN iar sfîrşitul datelor pe cartele este determinat de cartela EOF şi anume

```
.        RUN
          {cartele de date
.EOF
.        EOJ
```

În cazul în care dorim să rulăm aceleaşi program cu mai multe seturi de date pe cartele iar rezultatele prelucrărilor le vom afişa la imprimantă se poate utiliza următoarea secvenţă

```
.        JOB
.        COMPILE COBOL
            program sursă COBOL pe cartele sau apelat dintr-o biblio-
            tecă sursă
.        LINK
.        RUN
         { set date 1
.EOF
.        FETCH LN : *4
.        RUN
         {set date 2
.EOF
.        EOJ
```

18.16 Dacă pentru rezolvarea unei probleme avem un program principal care apelează un subprogram, succesiunea cartelelor de comandă este următoarea:

```
·         JOB
·         COMPILE COBOL
        { cartele program principal COBOL (CALL SUBPRG
        { USING argumente)
·         COMPILE COBOL
        { IDENTIFICATION DIVISION.
        { PROGRAM—ID. SUBPRG.
        { cartelele subprogramului SUBPRG (eventual copiate din
        { biblioteca sursă cu COPY)
·         LINK
        { cartele de comandă pentru fișierele prelucrate de program
        { și subprogram (INIT, DELETE, ALLOC, ASSIGN
        { LABEL)
·         RUN
          { set de date pe cartele
·EOF
·         EOJ
```

Observație: trebuie făcută distincția între executarea unui program sau a unei lucrări. Executarea unui program se realizează după ce au fost parcurse etapele de compilare și editare legături prin citirea unei cartele RUN. O lucrare începe prin citirea unei cartele JOB și se termină la citirea primei cartele EOJ și poate conține execuția unuia sau a mai multor programe, deci poate conține mai multe cartele RUN. În momentul în care la un program dintr-o lucrare apare o eroare care întrerupe execuția normală a acestuia, în continuare nu se vor mai executa programele lucrării respective, lucrarea fiind abandonată.

18.17 *Î.* Fazele de execuție ale unui program sînt declanșate de citirea din fișierul de lucrări al sistemului a cartelei . . . . . . pentru compilare, a cartelei LINK pentru . . . . și a cartelei RUN pentru . . . .

R. — COMPILE;
— editarea de legături;
— execuția programului.

18.18 *Î.* O lucrare începe prin citirea . . . . . și poate să conțină . . . . și se termină la citirea unei cartele . . . .

R. — unei cartele JOB;
— execuția unuia sau a mai multor programe;
— EOJ.

18.19 *Î.* Setul de cartele de date (fișierul pe cartele) este plasat după cartela de comandă . . . . iar sfîrșitul fișierului de date pe cartele este determinat prin citirea . . . .

R. — RUN;
— .EOF (End Of File).

18.20 *Î.* După cartela de comandă LINK se pot introduce cartele de comandă referitoare . . . . iar acestea trebuie să se găsească înaintea cartelei RUN care . . . .

R. — la fișierele prelucrate în program;
— declanșează execuția propriu-zisă a programului.

18.21 Cartelele de comandă se referă în general la prelucrarea unei lucrări ceea ce presupune încărcarea și lansarea în execuție a programelor în format IMT înregistrate într-o bibliotecă utilizator în format IMT sau în biblioteca utilizator standard (BUS).

18.22 Cartela JOB identifică începutul unei noi lucrări iar tot ce urmează pînă la întîlnirea cartelei EOJ (End Of Job) face parte din această lucrare.

Format

- [Etichetă] JOB Ident,AN: nnnn,PN; nume

unde:

- *etichetă* — etichetă opțională;
- *ident* — identificatorul (numele) lucrării care începe, compus din 1 la 8 caractere alfanumerice din care primul este obligatoriu o literă;
- *nnnn* — numărul de cont al lucrării, compus din 4 caractere alfanumerice și este utilizat la contabilizarea lucrărilor executate pe calculator;
- *nume* — numele programatorului (utilizatorului), compus din 1 la 8 caractere alfanumerice, în afară de spațiu.

18.23 **Exemplu**

- JOB PROGRAM1,AN : CCO1,PN : MARIANA

numele lucrării este PROGRAM1, numărul de cont CCO1 iar numele programatorului MARIANA.

18.24 *Î.* Cartela JOB identifică . . . . și specifică pentru aceasta . . . .
și . . . .

R. — începutul unei noi lucrări;
— numele lucrării, numărul de cont și numele programatorului (utilizatorului).

18.25 *Cartela RUN* provoacă lansarea în execuție a unui program pentru care s-au executat etapele compilare și editarea legăturii și se găsește în fișierul de ieșire al editorului de legătură (fișierul LOADGO) sau care a fost apelat dintr-o bibliotecă în format IMT cu o cartelă FETCH.

Format simplificat:

- Etichetă RUN [FN : ident] [,NL : nr.1] [,TIME : nr.2]

$$\left[, \left\{ \begin{matrix} \text{UN} \\ \text{OD} \\ AD \end{matrix} \right\} : \left\{ \begin{matrix} CP \\ \\ \text{Adresa1, Adresa2} \end{matrix} \right\} \right]$$

• *ident* — identificatorul programului și este opțional în cazul în care cartela RUN este precedată de una din cartele de LINK sau FETCH;
• *nr. 1* — numărul maxim de linii de imprimat, număr zecimal și nu trebuie să depășească 524288, dacă lipsește se consideră 1000;
• *nr. 2* — timpul maxim de execuție a programului, este un număr zecimal de 3 cifre iar în lipsă se consideră implicit 3 minute după care dacă lucrarea nu s-a terminat este abandonată;
• *UD* — indică afișarea la imprimantă a memoriei corespunzătoare programului pentru sfîrșitul normal al acestuia;
• *OD* — indică afișarea memoriei în cazul sfîrșitului anormal și este posibilă la cererea operatorului, după apariția mesajului DUMP la consolă operatorul poate răspunde Y(da) sau N(nu);
• *AD* — afișarea memoriei pentru sfîrșitul anormal al programului;
• *CP* — afișarea completă a zonei de memorie ocupată de program;
• *Adresa1*, *Adresa2* — reprezintă limitări ale afișajului de memorie;

18.26 *Exemplu:*
• RUN UD:CP

Executarea acestei cartele va produce afișarea zonei de memorie corespunzătoare programului pentru sfîrșitul normal al acestuia. Prin absența parametrilor TIME și NL se consideră timpul de execuție 3 minute iar numărul de linii la imprimantă 1000;
• RUN FN:PROGRAM1,TIME:15,NL:3000

Această cartelă produce încărcarea în memorie și lansarea în execuție a programului PROGRAM1 care se găsește în BUS. Timpul de execuție afectat este de maxim 15 minute iar numărul de linii afișate la imprimantă este de maxim 3000.

18.27 *Î.* Cartela RUN produce . . . . are formatul obligatoriu . . . . . iar în lipsa argumentelor NL și TIME valorile implicate sînt . . . . și . . . .

R. — lansarea în execuție a unui program;
— • RUN;
— 1000 de linii afișate la imprimantă și 3 minute timpul maxim afectat pentru execuția programului.

18.28 Cartela EOJ anunță sfîrșitul unei lucrări
Format
• [Etichetă] EOJ [Ident]

Această cartelă permite monitorului să execute reinițializările necesare începerii unei noi lucrări, care va fi anunțată cu o nouă cartelă JOB.

18.29 *Î.* O lucrare începe să se execute la întîlnirea cartelei . . . și se termină la citirea . . . .

R. — JOB;
— cartelei EOJ.

18.30 *Î.* Pentru compilarea unui program COBOL sînt necesare următoarele cartele de comandă:

. . . .
. . . .
program sursă COBOL
. . . .

R. JOB
. COMPILE COBOL
program sursă COBOL
. EOJ

18.31 *Î.* Ce cartele de comandă sînt necesare pentru execuția aceluiași program și care este succesiunea lor știind că în program este prelucrat un fișier pe cartele?

R. . JOB
. COMPILE COBOL
*
* program sursă COBOL
*
. LINK
. RUN
{Cartele de date}
. EOF
. EOJ

18.32 Cartela FETCH permite căutarea și încărcarea în memorie a unui program, ce se găsește într-o bibliotecă utilizator în format IMT sau în fișierul de ieșire al editorului de legături.

Format:

. [Etichetă] FETCH LN:nume.bibl [,DV:periferic] [, { CD:data-creare | GN:nr. generație }] [,VN : nr. versiune] [,FN:Ident] [,UN:nr-punere-la-zi]

18.33 Argumentele au următoarele semnificații:

• *nume-bibl* — numele bibliotecii care conține programul Ident și este format din 1 la 8 caractere alfanumerice. Dacă programul se găsește în fișierul de ieșire al editorului de legături (fișierul LOADGO) ca urmare a unei editări de legături precedente atunci *nume-bibl* ia valoarea ∗4 și singurul parametru obligatoriu este LN;

• *periferic* — precizează perifericul pe care se găsește biblioteca IMT, este definit în formatul cartelei ASSIGN;

• *data-creare* — data de creare a bibliotecii, din cinci cifre (aazzz);

• *nr-generație* — numărul de generare al bibliotecii, din patru cifre; Dacă CD și GN lipsesc se consideră valoarea zero.

• *nr-versiune* — numărul de versiune al bibliotecii și este un număr zecimal de două cifre (00 la 99);
• *Ident* — identificatorul programului apelat din biblioteca IMT;
• *nr-punere-la-zi* — numărul de actualizare a programului respectiv în bibliotecă și este un număr zecimal cu valoarea cuprinsă între 1 și 255.

Exemple:

. FETCH LN : BIBIMT,DV : AD1,FN : PROGRAM1,GN:1,VN:1

La întîlnirea acestei cartele programul PROGRAM1 din biblioteca BIBIMT care se găsește pe unitatea de discuri AD1, va fi încărcat în memorie după ce se vor verifica argumentele GN și VN;

. FETCH LN : * 4

Această cartelă se referă la încărcarea în memorie a unui program care se găsește în fișierul de ieșire al editorului de legături.

18.34 Î. O cartelă FETCH produce . . . .
argumentul 'LN : nume-bibl' reprezintă . . . .
iar argumentul 'FN : Ident' specifică . . . .

R. — căutarea și încărcarea în memorie a unui program memorat într-o bibliotecă IMT;
— numele bibliotecii de utilizator în format IMT;
— numele programului ce se apelează din bibliotecă.

18.35 Î. Ce cartele de comandă sînt necesare și care este succesiunea lor pentru apelarea și lansarea în execuție a unui program memorat într-o bibliotecă IMT?

R. . JOB
. FETCH
{cartele de comandă referitoare la fișierele preluate în program
. RUN
. EOJ

18.36 Cartelele de comandă pentru programele de traducere și editorul de legături se referă la punerea în lucru a programelor de traducere și a editorului de legături ce se găsesc memorate în BUS și care acceptă ca date de intrare programe de utilizator (în format sursă sau BT) iar rezultatele acestor prelucrări pot fi catalogate în biblioteci de utilizator, perforate pe cartele sau trecute în etapa următoare de prelucrare în vederea execuției finale. Aceste opțiuni asupra execuției unui program rezultă din argumentele cartelelor de comandă atașate acestuia.

18.37 La întîlnirea cartelei COMPILE este încărcat în memorie și lansat în execuție un program de traducere (compilator COBOL, FORTRAN, ASSIRIS etc.) ce se găsește în BUS. Ca intrare un program de traducere admite modulul sursă (program pe cartele în fișierul de lucrări sau program sursă apelat dintr-o bibliotecă sursă). Rezultatul obținut în urma compilării se numește modul obiect și poate conține următoarele tipuri de module obiect: modul de program (P), modul de date (D), modul de date comune (C).

Acestea sînt păstrate pe discul sistem constituind fişierele FILEDIT şi REPEDIT.

Format :

· [Etichetă] COMPILE COBOL [,opt 1] . . . [,opt n]
[,FN : Ident,Ln : nume-bibl,DV : periferic]

$\left[,\left\{\begin{matrix}\text{CD : Data-creare}\\ \text{GN : nr-generaţie}\end{matrix}\right\}\right]$, [VN :-nr-versiune]

18.38 Opţiunile asupra compilării sînt facultative şi au semnificaţiile următoare :

· *SEQ* — (SEQuenced) — produce controlul numărului de secvenţă (col-1-6) al cartelelor sursă. Secvenţa trebuie să fie crescătoare, în caz contrar cartela va fi tipărită cu caracterul S în faţă, iar controlul se reia luînd numărul de secvenţă al acestei cartele ca nouă bază de referinţă. La cartelele unde numărul de secvenţă lipseşte controlul nu se efectuează ;

• *NLG* — (No-ListinG) — prezenţa acestei opţiuni suprimă afişarea la imprimantă a textului sursă ;

• *DBG* — (DeBuG) — această opţiune provoacă tratarea de către compilator a cartelelor de punere la punct — cartele asemănătoare cu cartelele COBOL dar conţin un caracter „+" în coloana 7 iar semnul „/" în coloana 7 permite continuarea unei cartele cu "+". Dacă această opţiune este absentă, cartelele cu "+" sînt tratate ca comentarii,imprimarea lor este precedată de un caracter C ;

• *OBL* — (OBject Listing) — produce imprimarea programului obiect produs de compilator şi a zonelor rezervate şi iniţializate pentru diferite date, constante, tabele ;

• *MAP* — prezenţa opţiunii MAP provoacă imprimarea a trei liste :
— geografia datelor ;
— geografia programului obiect, în lipsa opţiunii OBL ;
— geografia fişierelor ;

• *CRF* — (Cross ReFerence) — permite obţinerea tabelelor de referinţă pentru identificatorii utilizaţi, nume de date, de secţiuni şi paragrafe ;

• *CNV* — (CoNVersions) — produce imprimarea diferitelor conversu utilizate, a operanzilor acestora şi liniile de program unde ele sînt apelate. Afişarea se face sub forma unor mesaje în lista mesajelor de erori şi au nivelul de gravitate zero.

• *SRG* — (Subscript — RanGe) — antrenează generarea secvenţelor de control a valorilor indicilor utilizaţi, astfel încît dacă un indice în timpul execuţiei ia o valoare superioară se va imprima la consola sistemului mesajul :

'O1-EN adresă-absolută-în-memorie, DEBORDEMENT DE L'INDICE : valoarea — indicelui' iar utilizatorul poate răspunde S (stop) şi execuţia se întrerupe sau G (go) şi execuţia continuă cu indicele eronat. Utilizînd programul obiect listat (Opţiunea OBL) şi avînd adresa de memorie din mesaj se poate identifica enunţul de program unde se produce depăşirea.

Prezenţa acestei opţiuni provoacă controlul intrărilor/ieşirilor şi a deschiderii — închiderii fişierelor sistemului, producînd apariţia mesajului :

$$*\left\{\begin{array}{c}1\\2\\3\end{array}\right\}\left\{\begin{array}{l}\text{E/S SUR FICHIER NON OUVERT}\\\text{OPEN SUR FICHIER OUVERT}\\\text{CLOSE SUR FICHIER FERME}\end{array}\right\}$$

unde : *1 — fişierul de intrare al sistemului (de obicei cititorul de cartele) ;
*2 — fişierul de ieşire al sistemului (de obicei imprimanta) ;
*3 — fişierul de perforare al sistemului utilizatorul are posibilitatea să răspundă de la consola sistemului cu S (Stop) sau G (GO) avînd semnificaţiile anterioare ;

• *PUN*—(PUNch) — antrenează la sfîrşitul compilării perforarea programului obiect în binar translatabil. Pachetul de cartele astfel obţinut poate fi supus unei editări de linii şi executat ;

• *DLG*—(Diagnostic LanGuage) — această opţiune permite alegerea limbii în care vor fi listate mesajele de erori. Aceste opţiuni sînt FRA (limba franceză), ROM (limba română) ; GER (limba germană). În lipsă se listează în limba fixată la generarea sistemului ;

• *DTB*—(Diagnostic TaBle) — această opţiune produce listarea tuturor mesajelor de eroare pe care le conţine compilatorul în limba indicată de parametrul DLG ;

• *ER : n*, această opţiune permite să se indice prin n(0÷4) gradul de eroare peste care nu se admite trecerea la faza de prelucrare ce urmează compilării, editarea de legături. În lipsa acestei opţiuni se consideră prin lipsă n = 2 (vezi lecţia 16).

Observaţie : — catalogarea rezultatului compilării COBOL nu se efectuează dacă erorile apărute nu au un nivel inferior celui admis. Dacă n = 3 catalogarea se poate executa numai cînd programul are erori de gravitate 2, deoarece pentru cele de gravitate 3 nu se traduce secvenţa sursă ;

• *Ident* — reprezintă numele atribuit modulului obiect, rezultat în urma compilării, cu care va fi memorat în bibliotecă şi poate avea maxim 6 caractere alfanumerice în afară de spaţiu iar primul obligatoriu să fie o literă.

18.39 *Î.* Citirea unei cartele COMPILE produce . . . . care acceptă ca intrare . . . . în rezultatul compilării se numeşte . . . .

R. — încărcarea în memorie şi lansarea în execuţie a unui program de traducere ;
— modulul sursă pe cartele sau apelat dintr-o bibliotecă sursă ;
— modul obiect.

18.40 *Î.* În formatul cartelei COMPILE COBOL opţiunea SEQ provoacă . . . ., opţiunea DBG are ca rol . . . . iar DLG specifică . . . .

R. — controlul numărului de secvenţă al cartelelor sursă;
— tratarea cartelelor de punere la punct a programului;
— limba în care vor fi listate mesajele de eroare.

18.41 *Cartela de comandă LINK* încarcă editorul de legături şi îl lansează în execuţie. Această fază acceptă ca intrare rezultatul compilării şi anume modulul obiect. Rezultatul etapei de editare legături se găseşte în fişierul LOADGO şi este un program executabil, care poate fi executat (utilizînd cartela RUN) sau/şi catalogat într-o bibliotecă utilizator în format IMT sau BUS.

Format:

- [Etichetă] LINK FN : Ident [,LN : nume [,DV : periferic]

$$\left[, \begin{Bmatrix} \text{CD : data-creare} \\ \text{GN : nr-generaţie} \end{Bmatrix} \right] \text{ [,VN : nr-versiune]}$$

$$\text{[,PUN] [,ER : n] [,NSLB] [,XREF] [,LDEF] } \left[, \begin{Bmatrix} \text{FMS} \\ \text{NFM} \end{Bmatrix} \right]$$

Argumentele sînt opţionale şi au semnificaţiile:

- *Ident* — numele programului în format IMT, maxim 8 caractere alfanumerice diferite de spaţiu iar primul caracter să fie o literă;
- *Nume* — numele bibliotecii utilizator în format IMT;
- *PUN* — prezenţa acestui parametru indică faptul că programul în format IMT se perforează pe cartele şi solicită prezenţa în cartela LINK şi a argumentului FN;
- *ER: n* — prin n se indică nivelul maxim de eroare acceptat, în lipsa parametrului ER se subînţelege n = 1. Erorile sînt clasificate în 5 grupe şi anume:

0 — nu sînt erori;
1 — erori posibile în faza de execuţie;
2 — s-a detectat o eroare pentru care editorul de legături a ales o soluţie posibilă;
3 — eroare ireparabilă de către editorul de legături;
4 — eroare care nu permite încărcarea programului în memorie.

- *NSLB* — prezenta acestui parametru indică faptul că biblioteca de subprogr ame standard nu este accesibilă în această fază;
- *XREF* — apariţia acestui parametru solicită afişarea unei liste cu adresele identificatorilor;
- *LDEF* — se referă la afişarea listei punctelor de intrare şi a adreselor acestora;
- *FMS* — acest parametru precizează că este solicitată încorporarea în program a modulelor SGF, în lipsă este subînţeles;
- *NFM* — prezenţa acestui parametru interzice încorporarea în program a modulelor SGF, acestea fiind încorporate ulterior într-o nouă editare de legături

18.42 Î. Citirea cartelei de comandă LINK produce începerea . . . . și acceptă ca intrare . . . .

R. — fazei de editare de legături;
— modulul obiect obținut în urma compilării.

18.43 Î. Rezultatul fazei editare de legături este . . . . care poate fi catalogat . . . . sau poate fi executat dacă este citită în continuare cartela de comandă . . . .

R. — un program executabil;
— într-o bibliotecă în format IMT;
— RUN.

18.44 *Exemplu practic:*

Pentru obținerea unei compilări, editări de legături și a execuției unui program sursă în COBOL este necesară următoarea succesiune de cartele de comandă:

```
.    JOB PROGRAM1,AN:CC01,PN:ANDRA
.    COMPILE COBOL,SEQ
   {pachet de cartele cu  | SELECT FCART ASSIGN TO SYSIN.
   {programul sursă       | SELECT FLIST ASSIGN TO SYSOUT.
.    LINK
.    RUN NL:15000,TIME:10
   {set de cartele de date
.:EOF
.    EOJ
```

18.45 Cartela JOB anunță începutul unei lucrări, iar cartela următoare indică faptul că urmează un program sursă scris în COBOL și produce încărcarea în memorie a compilatorului COBOL. În timpul compilării se va face și controlul asupra secvențelor cartelelor program (coloanele 1 la 6) deoarece este prezent parametrul SEQ. Dacă în timpul compilării nu au fost detectate erori de grad mai mare ca 1, rezultatul compilării (modulul obiect) se trece în fișierele FILEDIT și REPEDIT. În caz contrar (există erori de grad 2, 3) execuția programului PROGRAM1 se întrerupe, sînt citite toate cartelele pînă la cartela EOJ și se trece la execuția unei noi lucrări, cea a programului PROGRAM2 din următoarea cartelă de JOB. Dacă în urma compilării nu au apărut erori care să oprească execuția programului, atunci aceasta continuă cu citirea cartelei LINK care produce încărcarea editorului de legături în memorie, în locul compilatorului COBOL și se trece la prelucrarea datelor aflate în fișierele de legătură dintre compilator și editorul de legături.

La fel ca și în cazul compilării, dacă în editarea de legături nu apar erori care să oprească execuția programului, în continuare se trece la citirea următoarei cartele de comandă, cartela RUN. Această cartelă provoacă lansarea în execuție a programului PROGRAM1 obținut în urma editării de legături. În această execuție numărul de rînduri tipărite la imprimantă maxim admis este de 15000 iar timpul maxim al execuției este de 10 minute. După cartela RUN urmează setul de cartele de date cu care se va executa programul. Întîlnirea cartelei EOF anunță sfîrșitul fișierului pe

cartele, iar în mod normal, deoarece programul are un singur fişier de intrare (fişierul pe cartele) iar prelucrarea acestuia s-a terminat, următoarea cartelă de comandă este cartela EOJ care anunţă sfîrşitul lucrării începute la citirea cartelei:

· JOB PROGRAM1, ...

Observaţie: deşi programul COBOL conţine descrierile a două fişiere — un fişier pe cartele (FCART) şi un fişier la imprimantă (FLIST) nu există cartele de comandă referitoare la aceste fişiere deoarece în rubricile SELECT acestea au fost atribuite fişierului de intrare al sistemului (SYSIN) respectiv fişierului de ieşire al sistemului (SYSOUT) şi în acest caz atribuirile la aceste periferice au fost implicite.

18.46 *Î.* După terminarea etapei de compilare, următoarea etapă este . . . ., care este declanşată de citirea cartelei de comandă cu punct . . . . şi care este luată în considerare şi interpretată numai dacă . . . .

R. — editarea de legături a modulului obiect;
— LINK;
— în faza de compilare nu au apărut erori care să întrerupă execuţia programului.

18.47 *Î.* În cartela de comandă LINK argumentul 'FN: Ident' reprezintă . . . . argumentul 'LN: Nume' specifică . . . ., argumentul 'DV: periferic' indică . . . .
iar prezenţa acestor argumente produc . . . .

R. — numele programului în format IMT;
— numele bibliotecii utilizator în format IMT;
— tipul de periferic şi numărul perifericului ce conţine biblioteca;
— memorarea într-o bibliotecă IMT a programului executabil obţinut în urma etapei de editare legături.

18.48 *Î.* După terminarea fazei de editare legături, execuţia poate continua cu . . . . sau cu . . . . numai în situaţia în care în etapa de editare legături nu au apărut . . . .

R. — execuţia finală a programului executabil;
— memorarea (catalogarea) programului într-o bibliotecă IMT;
— erori care să facă imposibilă execuţia în continuare a programului.

18.49 În figura 18.1 sînt prezentate schematic etapele execuţiei unui program cu evidenţierea posibilităţilor de a începe execuţia fiecărei etape şi de trecere la etapa următoare sau de a reţine rezultatul etapei executate pe un suport magnetic sau pe cartele.

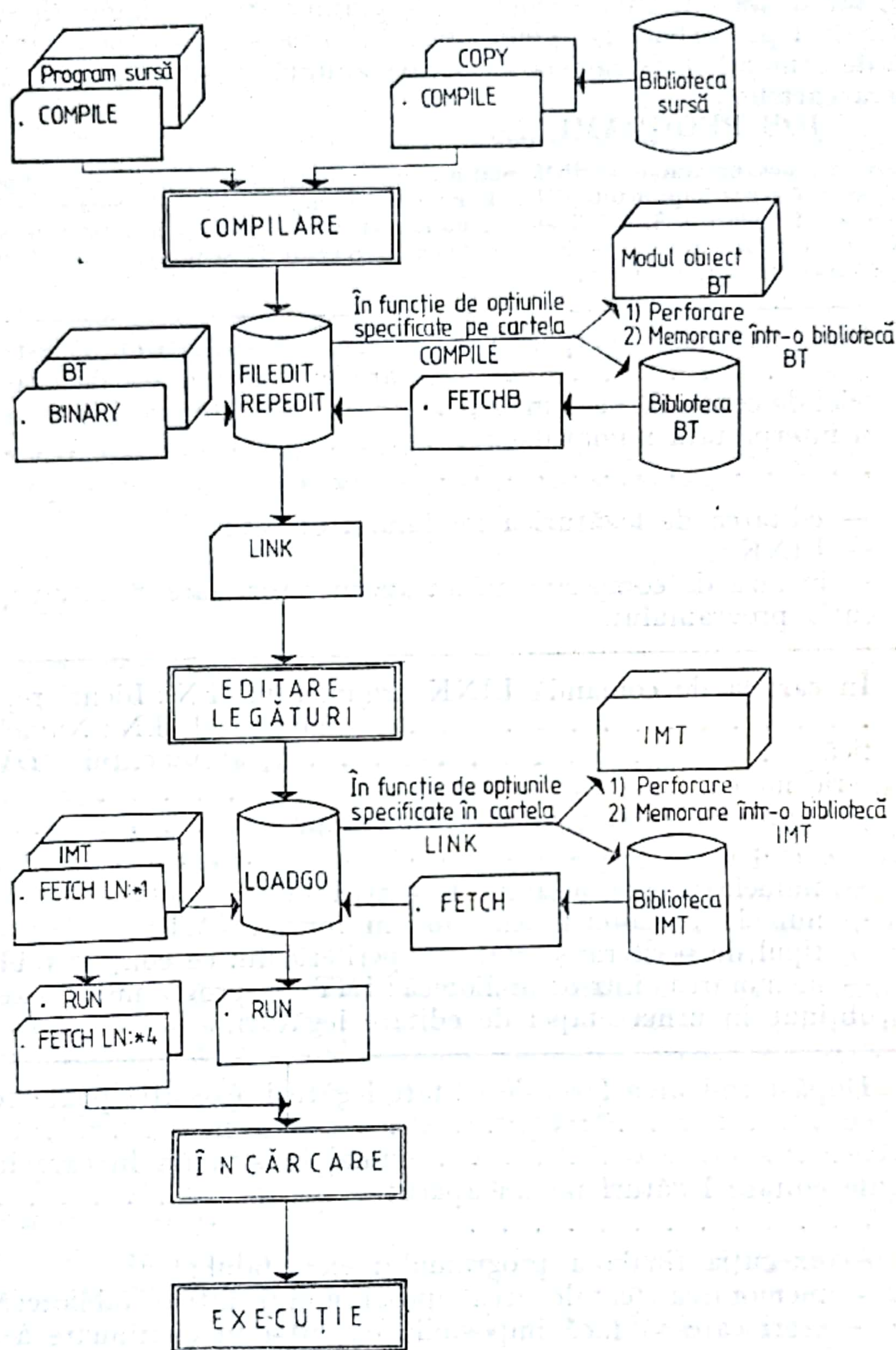


Fig. 18.1.

Observație : față de cartelele de comandă prezentate pînă aici în schemă apar cartelele BINARY și FETCHB pentru care nu vom prezenta descrierea formatului. Ele sînt utilizate pentru a anunța monitorul că următoarele cartele reprezintă un modul obiect în format BT

(cartela BINARY) sau că se dorește să se apeleze un program ce se găsește într-o bibliotecă utilizator în format BT (cartela FETCHB). În schemă mai apar notațiile:

*1 — numele simbolic, al fișierului de lucrări care poate fi pe cartele, bandă magnetică sau disc;

*4 — numele simbolic pentru fișierul de ieșire (LOADGO) al editorului de legături.

Sînt atribuite nume simbolice și pentru fișierul de la imprimanta sistemului (*2) și fișierului de la perforatorul de cartele al sistemului (* 3).

18.50 *Î.* Fazele execuției unui program sînt . . . .

R. — compilare, editare legături, încărcare cu execuție.

18.51 *Î.* Faza de compilare admite ca datele de intrare să fie sub forma . . . . sau utilizînd enunțul COPY acestea pot fi . . . . iar rezultatul acestei faze este prelucrat de . . . .

R. — pachet de cartele cu programul sursă (modul sursă);
— memorate într-o bibliotecă sursă;
— editorul de legături.

18.52 *Î.* Editorul de legături poate începe prelucrarea rezultatului obținut în etapa de compilare dacă . . . . iar rezultatul acestei prelucrări (etape) este . . . . care poate fi . . . .

R. — în compilare nu au apărut erori de un nivel care să producă oprirea execuției;
— un program executabil;
— încărcat și executat în continuare sau memorat (catalogat) într-o bibliotecă IMT.

18.53 Cartelele de comandă referitoare la prelucrarea fișierelor pe suporturi magnetice completează informațiile specifice în cadrul unui program pentru fișierele ce urmează a fi prelucrate. Informațiile referitoare la un fișier sînt prezentate pe mai multe tipuri de cartele de comandă. Astfel informațiile conținute de cartelele ASSIGN, LABEL, FILE se referă la un fișier individual pe suport și trebuie să conțină idex-ul atribuit fișierului în rubrica SELECT din program. Altă grupă de cartele se referă la suportul (volumul) pe care se găsește fișierul și ele nu conțin idex-ul fișierului, cum sînt cartelele ALLOC, INIT, DELETE.

Informațiile conținute în aceste cartele sînt necesare în faza de execuție a programului fiind obligatorie așezarea lor înaintea cartelei RUN.

Fișierele sistemului *1, *2, *3 nu necesită cartele de comandă referitoare la identificarea și afectarea lor, deoarece acesta se realizează automat prin specificarea în rubricile SELECT a idex-urilor SYSIN, SYSOUT SYSPUNCH.

18.54 *Î.* Cartelele de comandă referitoare la fișiere specifică . . . . legătura cu descrierea fișierelor din program se realizează prin . . . . iar aceste informații sînt prelucrate în faza . . . . și de aceea ele trebuiesc așezate înaintea . . . .

R. — informații referitoare la tipul și numărul perifericului, la numele fișierului etc;
— idexul atribuit fișierelor în rubricile SELECT;
— de execuție a programului;
— cartelei RUN.

18.55 *Cartela ASSIGN* permite afectarea unuia sau a mai multor periferice de același tip pentru un fișier, precizînd caracterul facultativ sau obligatoriu al afectării care se poate face pe durata lucrării sau a unei faze.

Format:

. [Etichetă] ASSIGN A, DV: perifN1 [sep perif N2] . . .
[sep perif Nn] [,HOLD]

Argumentele au semnificațiile:

• A — identificatorul de exploatare al fișierului (idex din rubrica SELECT);
perif N(1 ÷ n) reprezintă tipul și numărul perifericului;
perif — numele simbolic al perifericului, este compus din două caractere cu următoarea semnificație:

• RD-disc amovizibil (DIAM);
• FD -disc fix;
• MD—DIMAS (MD 25);
• AD—MD 50;
• BD—MD 100;
• CD—MD 200;
• MT—bandă magnetică;

DK -disc magnetic de orice tip;
PR—imprimantă;
CR—cititor de cartele;
CP—perforator de cartele;
TW—mașină de scris;
TR—cititor bandă perforată;

• N — număr zecimal cu valori între 0 și 99 și reprezintă numărul perifericului;

• sep — separator și poate fi unul din următoarele caractere '+' sau '/' cu semnificația:
+ semnifică afectarea obligatorie a perifericului al cărui nume simbolic și număr N urmează acestui separator;
/ semnifică faptul că afectarea se realizează dacă perifericul respectiv este disponibil;

• HOLD — acest parametru permite ca afectarea să fie conservată pe perioada întregii lucrări care poate avea mai multe faze de execuție, în absența acestui parametru afectarea este realizată pe durata execuției unui program din lucrare.

Observație: perifericele citate pe o cartelă ASSIGN trebuie să fie de același tip.

18.56 *Î.* Prin intermediul cartelei de comandă ASSIGN se afectează . . . ., afectare care se poate

realiza pe durata . . . . în funcție de prezența sau lipsa argumentului . . . . . din cartela respectivă.

R. — unul sau mai multe periferice de același tip pentru un fișier;
— lucrării sau a unei faze a acesteia;
— HOLD.

18.57 *Cartela LABEL* se referă la fișierele de pe suporturi reutilizabile (bandă magnetică și disc) și conține informații referitoare la identitatea fișierului creat sau exploatat și condițiile de afectare pentru suportul disc.

Format:

[Eticheta] LABEL A,FN : 'nume—fișier' [,GN : nr—generație] [,VN : nr—versiune]
[,FN : nn] [,UN : nr—punere—la—zi] [,CD : data—creare]
[,RP : rrr] [,AC : a] [,FS : ssssss] [,ON : 'nume—prop'] $\left[,AM:\begin{Bmatrix}ANY\\NEW\\OFL\end{Bmatrix}\right]$
[,SZ : ccccc] [,XU : xxx] $\left[,\begin{Bmatrix}OD:data—creare\\OG:nr—generație\end{Bmatrix}\right]$ [,OV : nr—versiune]
[,VP : 'cuvînt1'] [,FP : 'cuvînt2']

18.58 Argumentele au semnificațiile:

- *nume-fișier* — numele fișierului memorat în eticheta fișierului compus din 1 la 17 caractere alfanumerice;
- *nn* — reprezintă numărul de ordine al volumului în fișier și este un număr zecimal cu valori de la 1 la 99;
- *rrr* — perioada de reținere a fișierului, se specifică la creare și poate avea valori între 0 și 999 reprezentînd număr de zile;
- *a* — este un caracter diferit de spațiu și reprezintă codul de validare al procedurii de protecție prin cuvînt de trecere pentru accesul la fișier;
- *ssssss* — este utilizat și în cartela INIT pentru atribuirea unui nume comun pentru toate fișierele prezente pe volumul respectiv;
- *nume—prop* — *numele proprietarului*, format din 1 la 14 caractere alfanumerice iar în lipsă se consideră format din spații;
- *AM* — modul de afectare; acest parametru trebuie specificat cînd se dorește ca afectarea suportului fizic de pe disc să se efectueze de către SGF;
- *ANY* — alocarea se face pe oricare din volumele cu zone disponibile;
- *NEW* — se efectuează alocarea volumelor disponibile în întregime și se efectuează controlul asupra parametrilor FS și ON;
- *OFL* — indică afectarea pentru noul fișier a zonei aceluiași fișier de generație și versiune anterioară;
- *ccccc* — dimensiunea zonei de afectat în număr de cilindri, poate lua valori între 1 și 65535;

• *xxx* — dimensiunea zonei de extensie alocată fișierului dacă intervine o depășire de capacitate și poate avea valori între 1 și 199;

• *OD și OG* — reprezintă data de creare și numărul de generație pentru fișierul care se înlocuiește și se utilizează cînd se specifică AM: OFL;

• *OV* — numărul de versiune al fișierului care se înlocuiește AM: OFL) și este necesar de specificat cînd este diferit de zero;

• *cuvînt 1* — este format din 1 la 10 caractere alfanumerice și permite inițializarea unui volum protejat prin cuvînt de trecere la volum;

• *cuvînt2* — este format din 1 la 10 caractere alfanumerice și reprezintă cuvîntul de trecere la fișier, fiind obligatoriu dacă fișierul este protejat prin cuvînt de trecere.

Parametrii obligatorii ai cartelei LABEL sînt idexul fișierului (A) și numele fișierului (FN) ceilalți parametrii sînt facultativi și se face controlul asupra lor numai dacă sînt prezenți. Argumentele AM și SZ se utilizează numai la crearea fișierelor iar în rest sînt ignorate.

18.59 Î. Cartela de comandă LABEL este utilizată numai pentru . . . . iar argumentele obligatorii de specificat sînt . . . . și . . . .

R. — fișierele pe suporturi magnetice;
— identificatorul de exploatare al fișierului;
— numele fișierului.

18.60 Î. Prezența în cartela LABEL a argumentului RP: 75 semnifică . . . .

R. — cererea ca perioada de reținere a fișierului respectiv pe suport să fie de 75 de zile.

18.61 *Cartela INIT* servește pentru a inițializa sau premarca un volum de bandă magnetică sau disc. Această cartelă este utilizată cînd dorim să prelucrăm un volum nou, sau unui volum cu un conținut vechi (perimat) vrem să-i dăm o nouă utilizare. De asemenea se poate utiliza pentru a transforma un volum cu organizare nestandard într-un volum cu organizare standard. Inițializarea unui volum ce conține informații cu date de expirare nedepășită se poate face numai cu intervenția operatorului.

Format:

[Etichetă] INIT DV: perif N [,VS: vvvvvv] [,ON: 'nume—prop']
[,FS: ssssss] [,AC: a] [,VP: 'cuvînt1'] [,UL: xy]

18.62 Argumentele au semnificațiile:

• *perif N* — numele simbolic al perifericului, iar perif poate fi MT, RD, FD, MD, AD, BD, CD (vezi cartela ASSIGN);

• *vvvvvv* — reprezintă identificatorul atribuit volumului și este format din 6 caractere diferite de spațiu;

• *ON, FS, AC, VP* — vezi cartela LABEL;

• *xy* — este un parametru care se referă numai la benzile magnetice pentru prelucrarea etichetelor de utilizator (Users Label):

x — o cifră între 1 şi 9 indicînd rangul ultimei etichete de volum de utilizator (UVL) de creat, dacă x = 0 nu se crează nici o etichetă UVL;

y — un caracter dintre cifrele 1 la 9 sau literele A la Z indicînd rangul ultimei etichete de început de utilizator (UHL) de creat, dacă y = 0 nu se crează nici o etichetă UHL.

Observaţie: dacă DV este diferit de MT sau dacă VS este omis atunci parametrul UL este ignorat.

Etichetele UVL şi/sau UHL care se vor crea trebuiesc plasate după cartela INIT de exemplu:

```
·      INIT DV : MT3, VS : VOLUM1,UL : 1C
UVL1 . . . . . . . . . . . . . . . .
UHLA . . . . . . . . . . . . . . . .
UHLB . . . . . . . . . . . . . . . .
UHLC . . . . . . . . . . . . . . . .
```

18.63 *Î.* Cartela INIT este utilizată pentru . . . . iar argumentul VS : vvvvvv conţine . . . . şi este format . . . .

R. — a iniţializa sau premarca un volum de bandă magnetică sau disc;
— identificatorul atribuit volumului după iniţializare;
— din 6 caractere alfamerice diferite de spaţiu.

18.64 *Cartela ALLOC* se utilizează pentru a aloca o zonă a unui disc pentru un fişier. Alocarea se poate face la nivel de fiecare disc sau global pentru fişierele multivolum.

Format:

```
· Etichetă ALLOC DV : perifN1[{+ | /} perifN2] . . .,FN : 'nume-fişier'
      [,ON : 'nume—prop'] [,FS : ssssss] [,VP : 'cuvînt1']
            [,CD : data—creare]
            [,GN : nr—generaţie] [,VN : vv] [,RP : rrr] [,AC : a]
            [,SZ : ccccc]
            [,XU : xxx] [,AM : {ANY | NEW | SPC}]
            [,VL : vvvvvv/sss/[aaa] . . .]
```

Argumentele au semnificaţiile:

• *perif N1* — poate lua valorile AD, BD, RD, MD, FD (vezi cartela ASSIGN)

\+ — acest semn indică o montare în paralel a volumelor pe unităţile periferice (alocare în paralel);

/ — indică faptul că încărcarea volumelor se face în serie (alocare în serie);

• FN, ON, FS, VP, CD, GN, RP, AC, SZ, XU, AM cu opțiunile ANY și NEW au fost întîlniți și în cartelele LABEL și INIT și trebuiesc utilizați dacă au fost folosiți la inițializarea discului și vor conține aceleași valori;

• *SPC* — se referă la alocarea unor volume specifice; pentru fiecare volum utilizatorul specifică identificatorul volumului, dimensiunea de alocat și eventual adresa de început a zonei de alocat; dacă se utilizează acest argument se ignoră argumentul SZ;

• *VL* — definește volumul specific, dimensiunea și eventual adresa de început a zonei de alocat pentru cazul în care s-a specificat AM : SPC;

• *vvvvvv* — reprezintă identificatorul volumului atribuit la inițializarea discului;

• *sss* — este un număr zecimal între 1 și 199 și reprezintă numărul de cilindri pentru zona alocată fișierului;

• *aaa* — un număr zecimal între 1 și 199 și reprezintă adresa de început a zonei alocate fișierului.

Observație: volumele definite prin VL trebuie să fie menționate în ordinea montării lor indicată prin argumentul DV.

Alocarea este întreruptă la întîlnirea unui alt fișier cu aceeași identitate (nume-fișier, dată de creare, număr de generație, număr de versiune) pe unul din volume.

18.65 *Î.* Prin cartela de comandă ALLOC se poate afecta . . . . iar alocarea nu se poate realiza cînd pe discul respectiv . . . . sau . . . .

R. — o zonă de pe un disc pentru un fișier;
— se găsește un alt fișier cu aceeași identitate;
— nu mai este spațiu disponibil.

18.66 *Cartela DELETE* permite eliberarea zonelor de disc ocupate de un fișier dacă acesta este perimat, în caz contrar acesta se poate realiza numai cu intervenția operatorului.

Format:

$$[\text{Eticheta}]\,\text{DELETE DV : perif N1}\left[\left\{\begin{matrix}+\\/\end{matrix}\right\}\text{perifN2}\right]\ldots\text{,FN : 'nume-fișier'}$$

$$[\text{,ON : 'nume—prop'}]\ [\text{,VP : 'cuvînt1'}]$$

$$\left[,\left\{\begin{matrix}\text{CD : data—creare}\\ \text{GN : nr—generație}\end{matrix}\right\}\right]\quad [\text{,VN : vv}]$$

Argumentele din această cartelă au fost întîlnite în cartelele LABEL, INIT și ALLOC.

18.67 *Î.* Cu ajutorul cartelei DELETE se poate . . . . cu condiția ca . . . .

R. — șterge un fișier de pe disc;
— perioada de reținere a fișierului să fie expirată sau prin intervenția operatorului.

18.68 Î. Comparînd acţiunea cartelelor DELETE şi INIT prima realizează . . . . iar a doua . . . .

R. — eliberarea unei anumite zone de pe disc, afectate unui fişier; — realizează iniţializarea şi eliberarea întregului pachet de discuri.

18.69 Î. Un program de validare cartele, reţine cartelele corecte într-un fişier cu numele FISIER TRANZ pe o bandă magnetică iar pe cele eronate le afişează cu mesaje corespunzătoare într-o listă de erori. Rubricile SELECT pentru fişiere fiind următoarele:
SELECT FCART ASSIGN TO SYSIN.
SELECT FLIST ASSIGN TO SYSOUT.
SELECT FTRANZ ASSIGN TO AMT.
vă invităm să scrieţi cartelele de comandă ce însoţesc acest program.

R.
```
.       JOB VALID1,AN :CC11,PN :ANDRA
.       INIT DV : MT1, VS : 123456
.       COMPILE COBOL
        { program sursă COBOL
.       LINK
.       ASSIGN A,DV : MT1
.       LABEL A,FN : 'FISIER—TRANZ'
.       RUN NL : 3500,TIME : 15
        { set cartele de date
. EOF
.       EOJ
```
Observaţie: cartela . EOF indică sfîrşitul unui fişier pe cartele.

18.70 Î. Revenind la problema precedentă şi considerînd că fişierul ce reţine datele corecte va fi memorat pe un pachet de discuri vă invităm să precizaţi ce schimbări vor apare în cartelele de comandă şi în program.

R.
```
.       JOB . . . . . . . .
.       INIT DV : AD1,VS : 111111
.       ALLOC DV : AD1,FN :'FISIER—TRANZ',SZ : 15,AM :
        ANY
.       COMPILE COBOL
        { program sursă   SELECT FTRANZ ASSIGN TO AAD.
.       LINK
.       ASSIGN A,DV : AD1
.       LABEL A,FN : 'FISIER—TRANZ'
.       RUN NL : 3500,TIME :15
        { cartele de date
. EOF
.       EOJ
```

LECȚIA a 19-a ORGANIZAREA ȘI EXPLOATAREA BIBLIOTECILOR DE PROGRAME

- Formatele bibliotecilor
  - format sursa (SOU)
  - format binar translatabil (BT)
  - format imagine memorie translatabilă (IMT)
- cartele de comandă referitoare la programul bibliotecar
- operații cu biblioteci de programe
- proceduri catalogate

19.1 Organizarea și exploatarea bibliotecilor de programe este realizată de către programul bibliotecar.

Natura informațiilor conținute determină tipul de bibliotecă și anume:

I — bibliotecă în limbaj sursă;

II — bibliotecă în binar translatabil;

III — bibliotecă de programe în format IMT.

În general aceste biblioteci sînt organizate pe discuri însă cu anumite restricții se pot organiza și pe benzi magnetice. În continuare vom prezenta bibliotecile organizate pe discuri.

19.2 Un element al unei biblioteci se numește carte, iar conținutul cărții este caracteristic fiecărui tip de bibliotecă după cum aceasta este constituită din elemente în limbaj-sursă, în binar translatabil sau în format IMT. Astfel o carte conține:

I. una sau mai multe cartele în format sursă care ar putea reprezenta:

— descrierea unui fișier;

— descrierea unui raport;

— unul sau mai multe paragrafe din diviziunea de procedură;

— un program sursă;

— un subprogram sursă;

— proceduri catalogate;

II. rezultatul compilării unui program sursă care este constituit din modulul obiect program, modulul obiect de date și unul sau mai multe module obiect de comun;

III. un program IMT.

Programul bibliotecar poate acționa la nivelul unei biblioteci sau al unei cărți din bibliotecă.

Observație. Mai există un tip de bibliotecă și anume biblioteca de subprograme translatabile (RSL). Această bibliotecă se compune din subprograme în BT care sînt apelate de către programele de traducere sau editorul de legături în funcție de cerințele programului aflat în faza respectivă.

19.3 *Î.* O bibliotecă se compune din una sau mai multe . . . .
a) programe
b) module
c) cărți
R. c)

19.4 Programul bibliotecar poate crea și exploata două categorii de biblioteci:
— biblioteci standard (sursă, binar translatabil (RBN) și IMT);
— biblioteci de subprograme translatabile (RSL).

O bibliotecă standard se compune dintr-un fișier repertoar care conține pentru fiecare carte următoarele informații — numele cărții, numărul de punere la zi și adresa de pe disc a începutului fiecărei cărți — și zona în care sînt memorate cărțile bibliotecii.

Într-o bibliotecă toate cărțile conțin informații în același format.

19.5 O bibliotecă de subprograme în binar translatabil (RSL) conține subprograme utilizate frecvent de diferite programe ale sistemului sau de programe ale utilizatorului. Structura unei astfel de biblioteci este diferită de cea a bibliotecilor standard și ea oferă editorului de legături posibilitatea de regăsire rapidă a unui subprogram și de asamblare a lui într-un program IMT.

O bibliotecă de subprograme poate fi exploatată în două moduri diferite:
— prin intermediul bibliotecarului
— prin intermediul editorului de legături.

19.6 *Î.* O bibliotecă standard în funcție de tipul informațiilor conținute poate fi . . . . și . . . .
iar ca format se compune dintr-un . . . .
R. — sursă, binar translatabil (RBN) și IMT
— fișier repertoar și zona care conține cărțile bibliotecii.

19.7 *Î.* O bibliotecă RSL conține . . . . ce sînt apelate de . . . .
R. — subprograme în binar translatabil
— de editorul de legături pentru incorporarea lor în programe aflate în faza de editare de legături.

19.8 O bibliotecă în format sursă poate fi creată prin intermediul bibliotecarului.

Fiecare carte a bibliotecii este un fișier secvențial cu înregistrări de lungime fixă (80 octeți — imagine cartelă).

Lungimea unui bloc din biblioteca sursă este de 256 octeți.

Fișierul repertoar (BIBREP) conține informații referitoare la cărțile bibliotecii și are același format în bibliotecile sursă și binar translatabil (RBN).

Cărțile dintr-o bibliotecă sursă pot conține succesiuni de imagini cartelă într-o ordine oarecare. Nu se efectuează nici un control logic asupra conținutului unei cărți. Cartelele care conțin caracterul % în prima coloană nu sînt catalogate de către programul bibliotecar.

19.9 Î. Formatul de memorare în biblioteca sursă este . . . .

R. fix și corespunde unei imagini cartelă (80 octeți)

19.10 Î. O carte aparținînd unei biblioteci sursă conține . . . . asupra cărora nu se efectuează . . . . . .

R. — o succesiune de imagini cartelă într-o ordine oarecare
— nici un control logic.

19.11 O bibliotecă în format binar translatabil poate fi creată de către programul bibliotecar iar catalogarea de cărți în bibliotecă se poate realiza cu ajutorul programului bibliotecar sau a cartelei de comandă COMPILE care conține argumentele FN, LN, DV, CD (GN), VN.

19.12 Î. Utilizînd opțiunile cartelei COMPILE pentru catalogarea rezultatului compilării într-o bibliotecă binar transtalabil, în ce condiții se realizează această catalogare?

a) se verifică secvența cartelelor;
b) nu se efectuează nici o verificare;
c) nu se efectuează catalogarea dacă în compilarea modului sursă apar erori mai mari decît cele admise.

R. c)

19.13 Î. Catalogarea unui program într-o bibliotecă RBU se poate realiza folosind parametrii . . . . din cadrul cartelei . . . .

R. — FN, LN, DV, CD(GN), VN;
— COMPILE.

19.14 Programul în format IMT este produsul ultimei etape de prelucrări și este pregătit pentru a fi încărcat și executat sau memorat într-o bibliotecă IMT.

Este obținut la sfîrșitul etapei *editare de legături* pe fișierul LOADGO și poate fi catalogat într-o bibliotecă IMT dacă cartela LINK conține argumentele necesare acestei operații.

19.15 Într-o bibliotecă în format IMT există două zone:
— zona permanentă — conține cărți „permanente”
— zona temporară — conține cărți „temporare”

Fișierul repertoar al unei biblioteci IMT conține informații referitoare la numele și numărul de actualizare al cărții, adresa de început al primului bloc precum și informații referitoare la legăturile dintre cărți.

În fișierul repertoar informațiile referitoare la o versiune permanentă sînt urmate de informațiile referitoare la versiunile temporare ale acesteia.

19.16 Î. Versiunile în care se poate găsi un program într-o bibliotecă IMT sînt . . . . iar

versiunea care conține toate informațiile referitoare la program este . . . .

R. — versiunea permanentă și versiunea temporară
— versiunea permanentă

19.17 *Î.* La ștergerea versiunii permanente a unui program dintr-o bibliotecă IMT sînt șterse . . . .

R. — toate versiunile temporare legate de această versiune permanentă.

19.18 Nivelele de control ale programului bibliotecar sînt:

a) la nivel de bibliotecă — creare și/sau deschidere de bibliotecă; duplicare (copiere) a unei biblioteci; afișarea numelor cărților (repertoriului) unei biblioteci; compararea conținutului a două biblioteci;

b) la nivelul unei cărți dintr-o bibliotecă — introducerea unei cărți; înlocuirea unei cărți; schimbarea numelui unei cărți; actualizarea unei cărți; afișarea conținutului unei cărți; stergerea unei cărți dintr-o bibliotecă.

19.19 Utilizatorul poate preciza funcțiile și detalii de realizare a acestora de către programul bibliotecar prin intermediul cartelelor de comandă și a argumentelor acestora.

Cartelele de comandă referitoare la programul bibliotecar sînt de categoria a II-a și au următorul format:

% NUME Argument1 [,Argument 2] . . . [COMENTARII]
col. 1.

unde:

— Caracterul % — situat în coloana 1 anunță o cartelă de comandă din categoria a II-a;

— NUME — reprezintă numele (identificatorul) cartelei de comandă și începe obligatoriu din coloana 3 și după el urmează un spațiu;

— Argumentele — sînt într-un număr oarecare, în funcție de tipul cartelei de comandă și sînt separate prin virgulă;

— COMENTARII — sînt separate de argumente printr-un spațiu și pot ocupa restul cartelei.

19.20 Argumentele nu pot depăși coloana 71 iar în cazul continuării acestora pe cartela următoare, se introduce un caracter asterisc (*) în coloana 72 a cartelei care se continuă, pe următoarea cartelă se perforează % în coloana 1 iar descrierea argumentelor continuă din coloana 11 fără să se mai treacă numele cartelei din coloana 3-a. Continuarea se efectuează în așa fel încît să nu se producă continuarea unui argument pe două cartele.

Observație: Cartelele de comandă de categoria I-a sînt cele ce conțin în prima coloană caracterul punct ('.'), ele sînt prezentate în lecția 18.

19.21 *Î.* Nivelele de prelucrare la care se referă funcțiunile programului bibliotecar sînt . . . . și . . . .

R. — la nivel de bibliotecă
— la nivelul unei cărți din bibliotecă

19.22 *Î.* Funcțiunile dorite să fie executate de bibliotecar sînt precizate prin intermediul . . . . care au formatul . . .
. . . .

R. — cartelelor de comandă de categoria a II-a
— % în col.1 ; spațiu în col.2 ; numele cartelei de comandă în col.3 ; un spațiu obligatoriu între nume și argumentele cartelei respective·

19.23 *Î.* Argumentele unei cartele de comandă pot fi continuate pînă în col. . . . iar pentru a se preciza continuarea, în col. . . . se introduce caracterul . . . . .

R. — 71
— 72
— asterisc (*)

19.24 Anumite cuvinte cheie dintre argumentele cartelelor de comandă sînt construite din cîteva caractere, fiecare avînd o semnificație bine stabilită. Astfel :

O — (ORIGINE) biblioteca origine
D — (DESTINATION) biblioteca destinație
L — (LIBRARY)bibliotecă
FILE carte
N — (UPDATE) număr de actualizare

de exemplu :

OL — ORIGINE LIBRARY biblioteca origine
DL — DESTINATION LIBRARY biblioteca destinație
ON numărul de actualizare al unei cărți

19.25 Crearea și/sau deschiderea unei biblioteci se realizează cu ajutorul cartelei de comandă OPNLIB care furnizează bibliotecarului informațiile necesare și specifice bibliotecii ce urmează a fi prelucrate.

Format :

% OPNLIB idex, LN:nume—bibl,DV:tip—perif [,VN:nr—versiune]

$$\left[, \left\{ \begin{matrix} \text{GN : nr—generare} \\ \text{CD : data—creare} \end{matrix} \right\} \right], \text{ FT :} \left\{ \begin{matrix} \text{SOU} \\ \text{RBN} \\ \text{RSL} \\ \text{IMT} \end{matrix} \right\} [\text{,INIT}]$$

19.26 *Argumentele au următoarele semnificații :*

- *idex* — identificatorul de exploatare al bibliotecii și este o literă ;
- *nume—bibl.* — numele bibliotecii de creat și/sau deschis, se compune din 1 la 8 caractere alfanumerice iar primul trebuie să fie o literă ;
- *FT* — tipul bibliotecii ,este un argument obligatoriu ;
- *INIT* — utilizarea acestui argument este obligatorie numai la prima deschidere a unei biblioteci după care celelalte deschideri se efectuează fără acest parametru.

Observație : bibliotecilor ca la orice alt fișier pe disc trebuie să li se asocieze o cartelă ALLOC care să fie plasată înaintea cartelelor referitoare la programul bibliotecar. În această cartelă ALLOC prezența argumentului VN și a unuia dintre argumentele GN sau CD este obligatorie iar argumentul FN (numele bibliotecii — nume dat de utilizator) are o lungime de 11 octeți; împărțită în două zone :

— prima zonă conține numele bibliotecii aliniat la stînga în primii 8 octeți, dacă numele este mai mic de 8 caractere se completează cu spații ;

— a doua zonă conține unul din grupurile de caractere SOU RBN, IMT RSL indicînd tipul bibliotecii.

19.27 **Exemplu :** Crearea unei biblioteci sursă și a unei biblioteci IMT :

```
·    JOB CRBIBL,AN : CC01,PN : ANDRA
·    ALLOC DV : MD1,FN : 'BIBLS1 SOU',AM : ANY,SZ : 10,VN : 1,CD:79135
·    ALLOC DV : MD1,FN : 'BIBLIMT IMT', AM : ANY,SZ : 15,VN : 1,CD79135
·    SYSRUN BIBLIO
% OPNLIB A,LN : BIBLS1,DV : MD1,VN : 1,CD : 79135,FN : SOU,INIT
% OPNLIB B,LN : BIBLIMT,DV : MD1,VN : 1,CD : 79135,FN : IMT,INIT
% ENDLIB
·    EOJ
```

Observație : prin cartela · SYSRUN BIBLIO se încarcă spre execuție în memorie programul bibliotecar iar sfîrșitul execuției sale este anunțat prin cartela % ENDLIB.

19.28 *Î.* Pentru a utiliza biblioteca sursă BIBLS1 creată în 1927 ce argumente trebuie să conțină cartela OPNLIB care va produce deschiderea bibliotecii?

R. Va conține aceleași argumente în afara de argumentul INIT care nu trebuie specificat decît la crearea unei biblioteci.

19.29 *Î.* Ce cartele de comandă vor fi necesare pentru deschiderea bibliotecii IMT creată în exemplul 1927 și cu ce argumente?

R.

```
·    JOB PRELBIBL,AN : CCO1,PN : ANDRA
·    SYSRUN BIBLIO
% OPNLIB B,LN : BIBLIM1,DV : MD1,VN : 1,CD : 79135, FN :
·    IMT
     { prelucrări asupra bibliotecii și/sau asupra unei cărți din
       bibliotecă
% ENDLIB
·    JOB
```

Remarcă : pentru orice prelucrare asupra unei biblioteci aceasta trebuie să fie mai întîi deschisă prin specificarea cartelei OPNLIB.

19.30 Conținutul unei biblioteci poate fi copiat pe același suport sau pe un suport diferit, astfel o bibliotecă disc poate fi copiată pe bandă sau disc iar o bibliotecă bandă poate fi copiată pe altă bandă sau pe disc.

Copierea unei biblioteci se realizează cu cartela COPY.

Format :

% COPY DL : idex1,OL : idex2

idex1 — identificatorul de exploatare al bibliotecii de destinație ;
idex2 — identificatorul de exploatare al bibliotecii origine.

Exemplu:

O operație frecvent utilizată asupra fișierelor de pe discuri este salvarea conținutului acestora pe un suport bandă magnetică după ce au fost realizate prelucrările dorite pentru a elibera pachetul de discuri fără riscul de a se distruge conținutul fișierului respectiv. Acest procedeu este utilizat și pentru biblioteci și anume:

```
·  SYSRUN BIBLIO
% OPNLIB M,LN : BDISC,DV : MD1,GN : 1,VN : 1,FT : SOU
% OPNLIB N,LN : BDISC,DV : MT3,GN : 1,VN : 1,FT : SOU,INIT
% COPY DL : N,OL : M
% ENDLIB
```

19.31 *Î.* Se dorește copierea bibliotecii BDISC de pe banda magnetică pe un pachet de discuri. Care vor fi cartelele de comandă și argumentele necesare acestei operații?

R.

```
·  ALLOC DV:MD1,FN:'BDISC SOU',AM:ANY,SZ:10,
   VN : 1,GN : 1
   SYSRUN BIBLIO
% OPNLIB A,LN : BDISC,DV : MD1,GN : 1,VN : 1,FT : SOU,
INIT
% OPNLIB B,LNB:DISC,DV:MT3,GN:1,VN:1,FT:SOU
% COPY DL : A,OL : B
      { prelucrări asupra și/sau cu conținutul bibliotecii copiată
        pe MD1
% ENDLIB
·   OB
```

19.32 Cu ajutorul cartelei REPERT se poate executa afișarea numelor cărților unei biblioteci utilizator sau numele subprogramelor împreună cu punctele lor de intrare conținute într-o bibliotecă de subprograme translatabile. În cazul bibliotecii utilizator în format IMT se obține și numele segmentelor conținute în fiecare carte iar prin precizarea numelui cărții sînt afișate numai segmentele din această carte.

Format:

% REPERT OL : idex [,OF : nume—carte [,ON : nr—actualizare]]

unde:

- *idex* — identificator de exploatare al bibliotecii;
- *nume—carte* — numele cărții IMT pentru care dorim să se afișeze numele segmentelor;
- *nr—actualizare* — numărul de actualizare al cărții IMT.

Exemplu: Se dorește să se afle conținutul unei biblioteci sursă iar pentru o bibliotecă IMT se dorește afișarea numelor segmentelor cărții PROGR1, bibliotecile se presupun pe disc.

```
·  SYSRUN BIBLIO
% OPNLIB A,LN : BIBLS1,DV : MD1,VN : 1,GN : 1,FN : SOU
% OPNLIB B,LN : BIBLIMT,DV : MD1,VN : 1,GN : 1,FN : IMT
% REPERT OL : A
% REPERT OL : B,OF,PROGR1
% ENDLIB
```

19.33 *Î.* Ce efect are întîlnirea unei cartele

% REPERT OL : M

și ce acțiune poate să ia programul bibliotecar considerînd că argu-

mentul FT din cartela OPNLIB citită deja ar fi avut oricare din valorile permise?
1 . . . .
2 . . . .
3 . . . .

R. 1. dacă FT : SOU sau FT : BT se afişează lista cărţilor;
2. dacă FT : RSL se afişează numele subprogramelor şi punctelor de intrare;
3. dacă FT : IMT se afişează lista cărţilor şi a segmentelor conţinute de fiecare carte.

19.34 Compararea conţinutului se referă la biblioteci, cărţi sau segmente. Această funcţiune este realizată cu scopul de a controla identitatea elementelor comparate. Compararea poate continua pînă la întîlnirea a maxim 10 divergenţe dacă se utilizează argumentul GO din cartela COMPARE.
Format:

% COMPARE DL : idex1 [,{DP : subprogram1 | DF : cartel [,DN : număr1] [,DS : segment1]}]

,OL : idex2 [,{OP : subprogram2 | OF : carte2 [,ON : număr2] [,OS : segment2]]},}

[,{GO | *STOP*}]

unde:
- *subprogram1, subprogram2* — numele subprogramelor de comparat (RSL);
- *carte1, carte2* — numele cărţilor de comparat;
- *număr1, număr2* — numele de actualizare ale cărţilor carte 1 şi carte 2;
- *segment1, segment2* — numele segmentelor de comparat;
- prezenţa argumentului GO produce continuarea comparării pînă la 10 divergenţe întîlnite, dacă este omis sau este indicat STOP la prima divergenţă întîlnită compararea se opreşte.

Exemplu: Să se compare cartea FTRANZ din biblioteca A cu cartea FTRANZ, din biblioteca B şi să continuie procesul dacă apar divergenţe.

% COMPARARE DL : B,DF : FTRANZ,OL : A,OF : FTRANZ,GO

Dacă s-ar cere să se compare cele două biblioteci A şi B atunci:

```
·     SYSRUN BIBLIO
% OPNLIB A,LN:BDISC1,DV:MD1,GN:1,VN:1,FT:SOU
% OPNBIB B,LN:BDISC2,DV:MD1,GN:3,VN:O,FT:SOU
% COMPARE DL:B,OL:A
·  ENDLIB
```

19.65 Inserarea unei cărţi într-o bibliotecă se poate realiza prin intermediul cartelei INCLUDE care apelează funcţia de inserare a cărţilor în bibliotecă.

Format:

%INCLUDE OL:{idex1 / *1} ,OF:nume—carte [,ON:număr1][,DL:idex2]

[,PROC][,{LEFT / *RGHT*}][,CHEK][,HH:număr2]

unde:

- *idex* — identificatorul de exploatare al bibliotecii care conţine cartea de înserat în noua bibliotecă;
- **1* — cartea de inserat în biblioteca se găseşte în fişierul de lucrări;
- *nume—carte* — numele cărţii în biblioteca origine şi numele noii cărţi introduse (pentru idex1) iar pentru *1 reprezintă numele sub care se catalogheaza pachetul de cartele;
- *număr1* — numărul de actualizare al cărţii în biblioteca origine;
- *idex2* — identificatorul de exploatare al bibliotecii în care se inserează cartela;
- *PROC* — se poate specifica pentru cărţile în format IMT pentru a indica catalogarea unui procesor;
- {*LEFT* / *RGHT*} — se referă la cărţile inserate într-o bibliotecă sursă pentru a specifica poziţia în care bibliotecarul va introduce numărul de secvenţă al liniilor de program (LEFT coloanele 1 la 6; RGHT coloanele 73 la 80);
- *CHEK* — este utilizat pentru biblioteca sursă şi indică să se verifice numerele de secvenţă ale cartelelor de înserat;
- *număr2* — numărul de actualizare al cărţii ce se inserează.

Exemplu: să se catalogheze în biblioteca sursă BSURS1 programul sursă PRG7 pe cartele, iar cartela obţinută să se numească la fel.

```
•     SYSRUN BIBLIO
%  OPNLIB A,LN:BSURS1,DV:MD1,VN:1,GN:1,FN:SOU
%  INCLUDE OL *1,OF PRG7,DL:A
   {cartelele programului sursă PRG7
%  ENDLIB
```

19.66 *Î.* Pentru cartea PRG7 introdusă în biblioteca sursă BSURS1 din exemplul din 19.35 cum se verifică secvenţa cartelelor şi în ce coloane se introduce acesta.

R. În absenţa argumentului CHEK bibliotecarul nu efectuează nici o verificare a secvenţei iar prin absenţă este considerat argumentul RGHT, ceea ce produce inserarea secvenţei în coloanele 73—80.

19.67 Înlocuirea unei cărţi dintr-o bibliotecă cu o carte aparţinînd unei alte biblioteci sau inserate în fişierul de lucrări se realizează prin cartela REPLACE.

Format :

% REPLACE DL : idex1,DF : nume—carte1 [,DV : număr1],OL : { idex2 | ∗1 }

[,OF : nume—carte2[,ON : număr2]] [,PROC][, { LEFT | *RGHT* } [,CHEK]]

unde :

- *idex1* — identificatorul de exploatare al bibliotecii în care se face înlocuirea cărții
- *nume-carte1* — numele cărții de înlocuit ;
- *număr1* — numărul de actualizare al cărții de înlocuit ;
- *idex2* — identificatorul de exploatare al bibliotecii în care se găsește cartea ce trebuie introdusă în biblioteca cu idex1 ;
- *∗1* — cartea ce trebuie introdusă se găsește în fișierul de lucrări ;
- *nume—carte2* — este numele cărții de introdus ;
- *număr2* — numărul de actualizare pentru nume—carte2.

19.38 Cartela RENAME conține noul nume al cărții.

Format :

% RENAME nume—carte

unde :

- *nume—carte* — noul nume al cărții

Exemplu : să se introducă cartea PROG1 din biblioteca BSURS1 în biblioteca BSURS2 în locul cărții PROG2 și să i se schimbe numele în PROG3

```
·     SYSRUN BIBLIO
% OPNLIB A,LN:BSURS1,DV : MD1,GN:1,VN:1,FT : SOU
% OPNLIB B,LN:BSURS2,DV : MD1,GN:1,VN:O,FT : SOU
% REPLACE DL:B,DF : PROGRAM2,OL:A,OFPROG1,ON:3
% RENAME PROG3
% ENDLIB
```

Observație : cartela RENAME trebuie să urmeze uneia cartele REPLACE sau INCLUDE

19.39 Cartela DISPLAY permite afișarea la imprimantă sau perforarea conținutului unei cărți aflate într-o bibliotecă utilizator.

Format :

% DISPLAY OL : idex, { SP : suprogram | OF : nume—carte } [,ON : număr] [,OS : segment]

[,OPTN] [, { *LIST* | PUN | LIST,PUN }]

unde :

- *idex* — identificatorul de exploatare al bibliotecii ;
- *subprogram* — numele subprogramului din biblioteca RSL ;
- *segment* — numele segmentului de afișat din biblioteca IMT ;
- *OPTN* — prezența acestui argument cere controlul formatului de editare ;

• *LIST* — este implicit şi provoacă afişarea la imprimantă a cărţii;
• *PUN* — prezenţa argumentului produce perforarea cărţii respective;
• *LIST,PUN* — produce afişarea la imprimantă şi perforarea la perforatorul de cartele.

Exemplu: să se afişeze la imprimantă cartea VERIF1 conţinută de biblioteca sursă BIBLSRS

```
.     SYSRUN BIBLIO
% OPNLIB OL:A,LN:BIBLSRS,DV:MD1,GN:1,VN:1,FT:SOU
% DISPLAY OL:A,OF:VERIF1
% ENDLIB
```

19.40 Ştergerea unei cărţi dintr-o bibliotecă este determinată de cartela DELETE care are formatul:

% DELETE DL: idex,DF: nume—carte [,DV: număr1[,CLR]]

unde:

• *idex* — identificatorul de exploatare al bibliotecii ce conţine cartea;
• *CLR* — dacă este prezent produce ştergerea tuturor cărţilor cu nume — carte şi al căror număr de actualizare este mai mic decît număr1.

19.41 Actualizarea unei cărţi memorate într-o bibliotecă sursă se realizează prin utilizarea cartelelor EDIT (apelează funcţia de actualizare pentru o bibliotecă sursă) şi MOD (specifică tipul modificărilor şi locul în care se execută acestea pe parcursul actualizării).

Cartela EDIT are formatul:

% EDIT OL: idex,OF: nume—carte [,ON: număr] [,ST: pas[,IN: valoare]]

$$\left[,\left\{\begin{matrix}\text{LEFT}\\ \underline{\textit{RGHT}}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\textit{RET}}\\ \text{DEL}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\textit{LIST}}\\ \text{NLST}\end{matrix}\right\}\right]$$

unde:

• idex, nume—carte, număr—identificatorul bibliotecii, numele cărţii de actualizat şi numărul de actualizare al cărţii înainte de actualizare care va lua valoarea număr + 1 după actualizare;
• *pas* — număr întreg zecimal şi reprezintă pasul de secvenţă pentru noua carte;
• *valoare* — numărul de secvenţă al primei cartele, este un număr de la 1 şi 6 sau 8 cifre zecimale după cum este specificat LEFT sau RGHT;
• *RET, DEL* — după actualizare se reţine şi cartea care a trebuit actualizată (RET) sau se poate cere ştergerea acesteia (DEL);
• *LIST, NLIST* — după actualizare este afişată la imprimantă cartea obţinută (LIST) sau se poate suprima această afişare (NLIST).

Cartela MOD are formatul:

% MOD număr1 [,număr2] [,CHEK]

unde:

• *număr1, număr2* — sînt numere zecimale referitoare la secvenţă; şi pot avea de la 1 la 6 sau 8 cifre după cum este specificat argumentul LEFT sau RGHT. Dacă este indicat numai număr1 aceasta precizează

secvenţa după care se vor insera cartelele de program care urmează după cartela MOD. Dacă este indicat şi număr2, cartela MOD indică suprimarea (ştergerea) cartelelor cu secvenţă între număr1 şi număr2;

• *CHEK* — se verifică secvenţa cartelelor de inserat iar la apariţia unei erori de secvenţă este abandonată actualizarea.

Exemplu: din biblioteca BDSURS1 să se şteargă cartea PROG5 iar cartea PROG3 să se actualizeze după cum urmează:

— să se suprime cartelele 13, 14, 15
— după cartela 37 să se adauge cartela:

```
        77 I1 PIC 99 VALUE ZERO.
```

— după cartela 93 să se adauge cartela:

```
       COMPUTE I1 = I1 + 1
     SYSRUN BIBLIO
  % OPNLIB M,LN : BDSURS1,DV :MD1,GN :1,VN :1,FT :SOU
  % DELETE DL : M,DF : PROG5,DN : 3,CLR
  % EDIT OL : M,OF : PROG3
  % MOD 13, 15
  % MOD 37
       77 I1 PIC 99 VALUE ZERO.
  % MOD 93
       COMPUTE I1 = I1 + 1
  % ENDLIB
```

19.42 *Î.* În cartela DELETE din exemplul din 19.41 au mai fost precizate anumite informaţii referitoare la cartea PROG5 şi la ştergerea ei din bibliotecă. La ce se referă aceste argumente (DN : 3,CLR)?

R. Prin DN : 3 s-a precizat să se şteargă cartea PROG5 cu numărul de actualizare 3 şi de asemenea şi celelalte cărţi cu nume PROG5 şi cu număr de actualizare mai mic decît 3.

19.43 *Î.* În cartela EDIT din acelaşi exemplu argumentele opţionale nu au fost precizate. Ce valori sînt atribuite în acest caz pentru acestea?

R. Numărul de actualizare (ON : număr) are valoarea 1; pasul secvenţei cartelelor (ST : pas) are valoarea 1; numărul de secvenţă al primei cartele este 00000001(IN : valoare); secvenţă cartelelor cărţii PROG3 este la dreapta (prin absenţă se consideră RGHT); după actualizare va fi reţinută cartea veche (RET) şi se va lista noua carte obţinută (LIST).

19.44 În executarea aplicaţiilor pe calculator sînt programe ce se rulează în aceeaşi succesiune, iar între două execuţii cartelele de comandă respective sînt identice sau au mici modificări (de exemplu: DV : MT1 la rularea 1 iar la rularea 2 va fi DV : MT3 deoarece MT1 este defect sau ocupat). În aceste condiţii se poate uşura exploatarea lucrărilor prin utilizarea procedurilor catalogate. Astfel cartelele de comandă referitoare la un lanţ de programe sînt catalogate într-o bibliotecă sursă de utilizator sau în biblioteca sistemului Z%PROC sub forma unei cărţi. Cînd se doreşte execuţia unui lanţ de programe cu cartelele de comandă memorate într-o bibliotecă sub forma unei proceduri catalogate acestea sînt apelate prin utilizarea unei cartele XPROC.

19.45 *Cartela XPROC* este citită din fişierul de lucrări al sistemului şi are ca efect deschiderea bibliotecii şi citirea cărţii ce conţine procedura catalogată al cărei conţinut este trecut în fişierul Z%SYSFT care va înlocui fişierul de lucrări al sistemului pînă la întîlnirea unei cartele EST. Asupra cartelelor de comandă ce se găsesc în Z%SYSFT se poate efectua modificarea valorii unor parametri, se pot adăuga, şterge sau înlocui cartele de comandă prin intermediul cartelei MOD şi în acest fel modificările afectează numai rularea respectivă (în bibliotecă nu sînt executate modificări).

Format :

- [Etichetă] XPROC { Nume [,UN : număr] / *,DV : perifN ,parametri din cartela LABEL } [,MOD] [,DEF]

unde :

- *Nume* — numele cărţii din biblioteca de proceduri catalogate ;
- *număr* — numărul de actualizare ;
- * — se specifică cînd procedura catalogată se află într-o bibliotecă de utilizator ;
- *perifN* — numele simbolic şi numărul periferícului pe care se află biblioteca ; parametrii din cartela LABEL — definesc biblioteca aşa cum sînt prezentaţi în cartela LABEL cu precizarea că argumentul FN va avea forma FN : 'NUMEBIBLSOU/NUMEPROC' aşa cum este descris pentru cartela ALLOC în paragraful 18.57. ;
- *MOD* — acest argument precizează că vor urma cartele de modificări ;
- *DEF* — indică utilizarea procedurilor „în STREAM"

Dacă argumentul 'UN : număr' este absent se ia în considerare cartea cu numărul de actualizare cel mai ridicat.

Argumentul MOD specifică că urmează cartele de modificare. Acestea se termină printr-o cartelă ENDMOD. Cartelele de comandă din procedura catalogată se termină cu o cartelă EST şi nu trebuie să conţină nici o cartelă JOB sau EOJ. Posibilităţile de utilizare a procedurilor catalogate sînt :

— proceduri "în stream" utilizate pentru testarea (verificarea) procedurilor. În acest caz procedura nu este catalogată ci urmează după cartela XPROC (ce conţine argumentul DEF) în fişierul de lucrări al sistemului ;

— proceduri catalogate şi apelate pentru execuţie aşa cum se găsesc în bibliotecă ;

— proceduri catalogate cu parametrii formali care oferă posibilitatea de a stabili anumite condiţii referitoare la rulare în momentul în care aceasta se efectuează.

19.46 Parametrii formali reprezintă zone substituabile din cartelele de comandă ale procedurii şi sînt delimitaţi (încadraţi) de caracterul &. Pot avea formele :

— & nume-parametru : şir—de—caractere &

unde :

- nume—parametru — reprezintă numele parametrului formal ;

• şir de caractere — o valoare iniţială a parametrului care poate fi modificată ulterior.

Exemplu:

```
·       ASSIGN A,DV : & PERIF : AD1&
```

dacă în momentul rulării nu modificăm acest parametru formal atunci fişierul cu idexul A va avea atribuit perifericul AD1 (&nume-parametru&).

Exemplu:

```
·       ASSIGN B,DV : & PERIFERIC &
```

în acest caz trebuie precizată valoarea parametrului formal PERIFERIC înainte de rulare, altfel nu se va atribui nici o valoare argumentului DV în momentul execuţiei.

19.47 Prin intermediul cartelei MOD se pot efectua modificările asupra cartelelor de comandă dintr-o procedură catalogată şi are formatul:

```
·          MOD  {număr1 [,număr2] [,CN :  număr3]}
↑          ↑    {listă—parametri—formali         }
col.1      col.11
```

unde:

• *număr1* — numărul de secvenţă al cartelei după care urmează să se insereze o secvenţă de cartele (dacă nu mai urmează număr2) sau secvenţa primei cartele de şters sau înlocuit dacă este specificat şi număr2;

• *număr2* — numărul de secvenţă al ultimei cartele de şters sau de înlocuit;

• *număr3* — este un număr zecimal care specifică numărul cartelelor care urmează după cartela MOD şi care nu sînt prelucrate de aceasta;

• *listă* — parametri—formali — se utilizează pentru atribuirea de valori parametrilor formali înainte de execuţie procedurii catalogate — (vezi 19.49.).

19.48 Să se creeze în biblioteca sursă BIBLSURS o procedură catalogată cu numele FLUXUNU:

```
·               SYSRUN BIBLIO
% OPNLIB A,DV : AD1,LN : BIBLSURS,VN : 1,GN : 1,FT : SOU
% INCLUDE DL : A,OL : X1,OF : FLUXUNU
                .
                .
                .
·               ASSIGN B,DV : & NUME1 &
                .
                .
                .
·               ASSIGN C,DV : & NUME2 : MT1 &
                .
                .
                .
```

ASSIGN M,DV: & NUME3: MT2 &

.

.

.

% ENDLIB

19.49 Se dorește să se apeleze procedura catalogată FLUXUNU și să se introducă următoarele valori:

— pentru NUME1 valoarea AD1
— pentru NUME3 valoarea MT3

```
. XPROC*, DV : AD1, AD : 1, VN : 1, FN'BIBLSURSSOU/FLUX-
  UNU' ,MOD
. MOD & NUME1 : AD1 &, & NUME3 : MT3 &
. ENDMOD
. EOJ
```

Observație: după cartelele MOD este obligatorie prezența unei cartele ENDMOD.

19.50 O procedură catalogată are cartelele numerotate de la 1 la 57. Se dorește ca pentru execuție să se efectueze următoarele modificări:

— să se adauge 5 cartele în locul cartelelor 13, 14, 15;
— să se suprime cartele 27 la 30
— să se adauge 3 cartele după cartela 57.

```
.    XPROC*, DV : AD1, . . . . . . . . . . . . . . . . . ., MOD
.    MOD 13,15
     cartela 1 |
     cartela 2 |
     cartela 3 | orice cartele în afară de MOD, ENDMOD, JOB, EOJ
     cartela 4 |
     cartela 5 |
.    MOD 27,30
.    MOD 57
     cartela 1 |
     cartela 2 | orice cartele în afară de MOD, ENDMOD, JOB, EOJ
     cartela 3 |
.    ENDMOD
```

19.51 *Î.* După efectuarea acestor modificări cum se vor schimba cartelele din procedura catalogată existentă în biblioteca sursă?

R. Aceste modificări nu se efectuează în cartea (ce conține procedura catalogată) din bibliotecă ci asupra conținutului cărții care a fost citit și transferat în fișierul Z%SYSFT.

19.52 Programele 19.1, 19.2, 19.3, 19.4, 19.5, 19.6, 19.7, 19,8. 19.9, 19.10, constituie exemple rulate pe calculator.

```
•           INIT DV:AD1,VS:654321
•           ALLOC DV:AD1,FN:'BIBLIO11SOU',AM:ANY,SZ:10,VN:0,GN:1
•           ALLOC DV:AD1,FN:'BIBLIO22IMT',AM:ANY,SZ:10,VN:0,GN:1
•           SYSRUN BIBLIO
BIBLIO    STARTED
% OPNLIB A,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU,INIT

% OPNLIB B,LN:BIBLIO22,DV:AD1,GN:1,VN:0,FT:IMT,INIT

% ENDLIB

•           EOJ
```

Programul 19.1. Crearea unei biblioteci sursă şi a uneia IMT

```
•           SYSRUN BIBLIO
BIBLIO    STARTED
% OPNLIB A,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU

% INCLUDE OL:*1,OF:PROGRAM1,DL:A

FICHIER CREE        NOM = PROGRAM1,NUMERO DE MISE À JOUR =

% ENDLIB

•           EOJ
```

Programul 19.2. Memorarea unui program în biblioteca sursă

```
•           COMPILE COBOL
COBOL     STARTED                    COBOL*ANS** 2, 6*CATIMT   *
    0001                COPY 'DV:AD1,FN:'BIBLIO11SOU/PROGRAM1'
                                                  ,GN:1,VN:0,UN:255 '.
B   0002 001010 IDENTIFICATION DIVISION.                1
B   0003 001020 PROGRAM-ID. PROGRAM1.                   2
B   0004 001030 ENVIRONMENT DIVISION.                   3
                           ⋮
B   0024 001220        CLOSE FIS1     STOP RUN.         23

FIN DE COMPILATION COBOL  PASSAGE A LA PHASE SUIVANTE

•           LINK FN:PROGRAM1,LN:BIBLIO22,DV:AD1,VN:0,GN:1
LINK      STARTED

NOUVEAU FICHIER CREE

NOM =PROGRAM1 ,NUMERO DE MISE A JOUR =  1
            EOJ
```

Programul 19.3. Apelarea programului din biblioteca sursă şi catalogare în biblioteca IMT

```
•           ALLOC DV:AD1,FN:'FIS1',SZ:10,AM:ANY
•           FETCH LN:BIBLIO22,DV:AD1,FN:PROGRAM1,GN:1,VN:0
•           ASSIGN A,DV:AD1
•           LABEL A,FN:'FIS1'
•           RUN
PROGRAM1  STARTED
```

Programul 19.4. Apelarea programului catalogat

```
•        SYSRUN BIBLIO
BIBLIO   STARTED
X OPNLIB M,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU

X EDIT OL:M,OF:PROGRAM1

X MOD 22,22

001210      DISPLAY DATA-CREARE '   S-A CREAT FIS1'                          22      EFFACEE

            DISPLAY 'S-A CREAT FISIERUL FIS1 IN CARE S-A INTRODUS O INREG              INSEREE

            'ISTRARE FICTIVA LA DATA = '  DATA-CREARE                                  INSEREE
LISTAGE DU FICHIER EDITE
                001010 IDENTIFICATION DIVISION.
                001020 PROGRAM-ID. PROGRAM1.
                001030 ENVIRONMENT DIVISION.
                001040 CONFIGURATION SECTION.
                001050 INPUT-OUTPUT SECTION.
                001060 FILE-CONTROL.
                001070        SELECT FIS1 ASSIGN TO AAD  ORGANIZATION INDEXED
                001080                  ACCESS SEQUENTIAL RESERVE NO RECORD KEY CHEIE.
                001090 DATA DIVISION.
                001100 FILE SECTION.
                001110 FD    FIS1       RECORDING F  LABEL RECORDS STANDARD
                                        BLOCK  CONTAINS 8 RECORDS.
                001120 01    INREG-FIS1.
                001130      02     CHEIE        PIC X(7).
                001140      02     DATA-CREARE    PIC X(6).
                001150      02     FILLER       PIC X(67).
                001160 PROCEDURE DIVISION.
                001170 P1. OPEN    OUTPUT FIS1.
                001180      MOVE   LOW-VALUE TO CHEIE   ACCEPT DATA-CREARE FROM DATE
                001190      WRITE  INREG-FUR2  INVALID KEY DISPLAY 'EROARE PRELUCRARE'
                001200             CLOSE FIS1   STOP RUN.
                            DISPLAY 'S-A CREAT FISIERUL FIS1 IN CARE S-A INTRODUS O INREG
                            'ISTRARE FICTIVA LA DATA = '  DATA-CREARE
                001220      CLOSE FIS1     STOP RUN.

 FICHIER CREE        NOM = PROGRAM1,NUMERO DE MISE A JOUR =   2

 X ENDLIB
```

Programul 19.5. Afișarea cărții PROGRAM1 din biblioteca BIBLIO11

```
•         SYSRUN BIBLIO
BIBLIO    STARTED
% OPNLIB M,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU
% REPERT OL:M

     N O M  D E  L A  B I B L I O T H E Q U E :     BIBLIO11
              N U M E R O  D E  V E R S I O N :     00
         N U M E R O  D E  G E N E R A T I O N :    0001
                                  F O R M A T :     SOU
   NOM DU FICHIER  :          NUMERO DE MISE A JOUR  :

                  PROGRAM1                            1
                  PROGRAM2                            1
                  FLUXUNU                             1

% ENDLIB
```

Programul 19.6. Afişarea numelor cărţilor din BIBLIO11

```
•         INIT DV:MT1,VS:123456
•         SYSRUN BIBLIO
BIBLIO    STARTED
% OPNLIB M,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU

% OPNLIB N,LN:BIBLIO11,DV:MT1,GN:1,VN:0,FT:SOU,INIT

% COPY DL:N,OL:M

NOUVEAU FICHIER CREE :    PROGRAM1 ,NUMERO DE MISE A JOUR :  1

NOUVEAU FICHIER CREE :    PROGRAM2 ,NUMERO DE MISE A JOUR :  1

NOUVEAU FICHIER CREE :    FLUXUNU  ,NUMERO DE MISE A JOUR :  1

% ENDLIB


TRAITEMENT DES BIBLIOTHEQUES SUR BANDES
```

Programul 19.7. Copierea BIBLIO11 DE PE DISC PE BANDA

```
•         SYSRUN BIBLIO
BIBLIO    STARTED
% OPNLIB A,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU

% INCLUDE OL:*1,OP:FLUXUNU,DL:A

FICHIER CREE       NOM = FLUXUNU ,NUMERO DE MISE A JOUR =   1

% ENDLIB

•          EOJ
```

Programul 19.8. Memorarea procedurii catalogate FLUXUNU

```
.          $YSRUN BIBLIO
BIBLIO     STARTED
% OPNLIB M,LN:BIBLIO11,DV:AD1,GN:1,VN:0,FT:SOU
% DISPLAY OL:M,OF:FLUXUNU

    N O M  D E  L A  B I B L I O T H E Q U E  :        BIBLIO11
             N U M E R O  D E  V E R S I O N  :        00
       N U M E R O  D E  G E N E R A T I O N  :        0001
                                   F O R M A T  :

NOM DU FICHIER  :   FLUXUNU         NUMERO DE MISE A JOUR  :      1
    01    .        ALLOC DV:AD1,FN:'DISCCART',SZ:10,AM:ANY             1
    02    .        ALLOC DV:AD1,FN:'FIS1',SZ:10,AM:ANY                 2
    03    .        FETCH LN:BIBLIO22,DV:AD1,FN:PROGRAM1,GN:1,VN:0      3
    04    .        ASSIGN A,DV:AD1                                     4
    05    .        LABEL A,FN:'FIS1'                                   5
    06    .        RUN                                                 6

% ENDLIB
```

Programul 19.9. Afişarea cărţii FLUXUNU

```
.           XPROC *,DV:AD1,VN:0,GN:1,FN:'BIBLIO11SOU/FLUXUNU'
.           ALLOC DV:AD1,FN:'DISCCART',SZ:10,AM:ANY                 1
.           ALLOC DV:AD1,FN:'FIS1',SZ:10,AM:ANY                     2
.           FETCH LN:BIBLIO22,DV:AD1,FN:PROGRAM1,GN:1,VN:0          3
.           ASSIGN A,DV:AD1                                         4
.           LABEL A,FN:'FIS1'                                       5
.           RUN                                                     6
PROGRAM1 STARTED

800207    S-A CREAT FIS1
.           * APPROCHE FIN PROCEDURE CATALOGUEE
.           EST
                                   FIN DE LA PROCEDURE CATALOGUEE
```

Programul 19.10. Apelarea procedurii catalogate FLUXUNU

LECȚIA a 20-a MODULARIZAREA ȘI STRUCTURAREA PROGRAMELOR COMPLEXE

- principiile proiectării structurate
- subprograme COBOL
- subprograme FORTRAN
- subprograme ASSIRIS
- segmentarea programelor

20.1 Programele mari, scrise „dintr-o bucată" sînt neeficiente. Căutînd să se elimine acest neajuns tehnicile moderne de realizare a programelor recomandă în unanimitate modularizarea, adică alcătuirea programelor din părți care se asamblează la un loc cu ajutorul compilatorului, bibliotecarului și editorului de legături. Avantajele care decurg din modularitate sînt multiple: fiecare dintre părți reprezintă o problemă mai simplă de transpus în program; testarea și depanarea vizează module bine delimitate deci va fi mai ușoară; repartizarea muncii pe programatori va fi mai uniformă și la nivelul fiecăruia; o parte însemnată din module se va putea folosi în alte aplicații; schimbările în aplicație vor însemna schimbarea unui modul sau a cîtorva module — schimbările într-un mare program au totdeauna urmări imprevizibile; documentația aferentă se realizează și se utilizează ușor; programele rezultate pot ocupa mai puțin loc în memorie datorită posibilității suprapunerii segmentelor; întreținerea programelor va putea fi făcută și de către alte persoane decît autorii inițiali; crește transferabilitatea programelor de pe un calculator pe altul și „portabilitatea", adică posibilitatea de a adapta programele la necesitățile unei alte unități decît cea pentru care a fost proiectată.

20.2 Conceptul de modularitate implică mai multe aspecte între care: un modul să nu depășească anumite dimensiuni; un modul să îndeplinească o singură funcție bine precizată; să existe o singură intrare și o singură ieșire; după terminarea prelucrărilor dintr-un modul se revine la locul de unde a fost apelat modulul; să nu existe decît un număr limitat de structuri standardizate pentru module, folosirea cărora să fie obligatorie.

20.3 Pentru a standardiza modularizarea programelor, a fost elaborată tehnica *proiectării structurate.* Destinată în primul rînd principiilor împărțirii unei lucrări pe module, ea este preocupată de funcțiunea fiecărui modul și nu de logica lor internă. Calitatea unei soluții este măsurată prin *simplitate* și *independența* modulelor.

Conform acestei tehnici descompunerea unui program poate urma cinci paşi :

1. Descrierea structurii algoritmului printr-un număr mic (3—10) de procese principale.
2. Se identifică fluxul de intrare şi fluxul de ieşire a datelor.
3. Se identifică punctul pînă la care fluxul de intrare poate fi urmărit (punctul 1) şi punctul de la care se constituie fluxul de ieşire (punctul 2).
4. Cele două puncte împart structura programului în trei secţiuni:
   — sursă date
   — transformare
   — distribuire date
5. Cele trei secţiuni sînt luate ca module.

Pentru cele prea mari se reiau cei cinci paşi de mai sus.

Operaţia se reia pînă se ajunge la module de circa 40—60 enunţuri executabile, dimensiune considerată optimă.

Criteriile cele mai importante în împărţirea pe module sînt :

— coeziunea internă maximă a fiecărui modul ;

— legături minime între module (de ex. volumul de date vehiculat între module).

20.4 Într-un program COBOL există două verbe prin care se asigură caracterul modular la nivelul prelucrării :

• CALL — pentru apelarea subprogramelor externe (scrise în COBOL, ASSIRIS sau FORTRAN)

• PERFORM — pentru apelarea procedurilor interne (secţiuni sau paragrafe din acelaşi program).

20.5 Verbul CALL are formatul :

CALL nume—program[**USING** identificator—1 [identificator—2] . . .]

unde :

• *nume—program* reprezintă fie numele unui subprogram în COBOL, FORTRAN sau ASSIRIS fie numele unei intrări într-un astfel de subprogram.

Identificatorii din dreptul opţiunii USING reprezintă numele datelor care se transferă către subprogram, respectiv în care subprogramul va transmite datele către programul apelant.

Revenirea din subprogram, după terminarea prelucrărilor, se face la enunţul imediat următor lui CALL.

20.6 *CALL* seamănă cu PERFORM doar că în cazul lui CALL se apelează întreaga diviziune PROCEDURE a unui alt program COBOL.

În subprogram după PROCEDURE DIVISION se va scrie USING şi eventualii parametri. Aceştia trebuie să fie aceiaşi cu cei din dreptul opţiunii USING din CALL, scrişi în aceeaşi ordine, avînd aceeaşi descriere dar nu neapărat aceleaşi nume. Parametrii se definesc în mod real (în secţiunile FILE sau WORKING—STORAGE) în programul principal şi fictiv (în secţiunea LINKAGE) în subprogram.

Subprogramul poate avea date proprii declarate în secţiunile sale FILE şi WORKING-STORAGE. Un subprogram poate chema la rîndul său un alt subprogram.

20.7 Î. 1. În cadrul programului 20.1 există . . programe COBOL, și anume programul principal . . . . . și subprogramul . . . .
2. Apelarea subprogramului se face prin verbul . . . . . iar după clauza . . . . urmează parametrii, numiți în programul principal . . . . iar în subprogram . . . .
3. Parametrii sînt definiți în programul principal în secțiunea . . . . iar în subprogram în secțiunea . . . . .
4. Care este primul enunț din programul principal executat după revenirea din subprogram?

R. 1. două, PRGM, SECVENTA
2. CALL, USING, REVENIRE, INREG, RETUR, INREG—L
3. WORKING—STORAGE, LINKAGE
4. PERFORM PRELUCRARE ...

20.8 Î. Poate exista CALL fără USING? Dacă da, ce implicații are aceasta pentru subprogram?

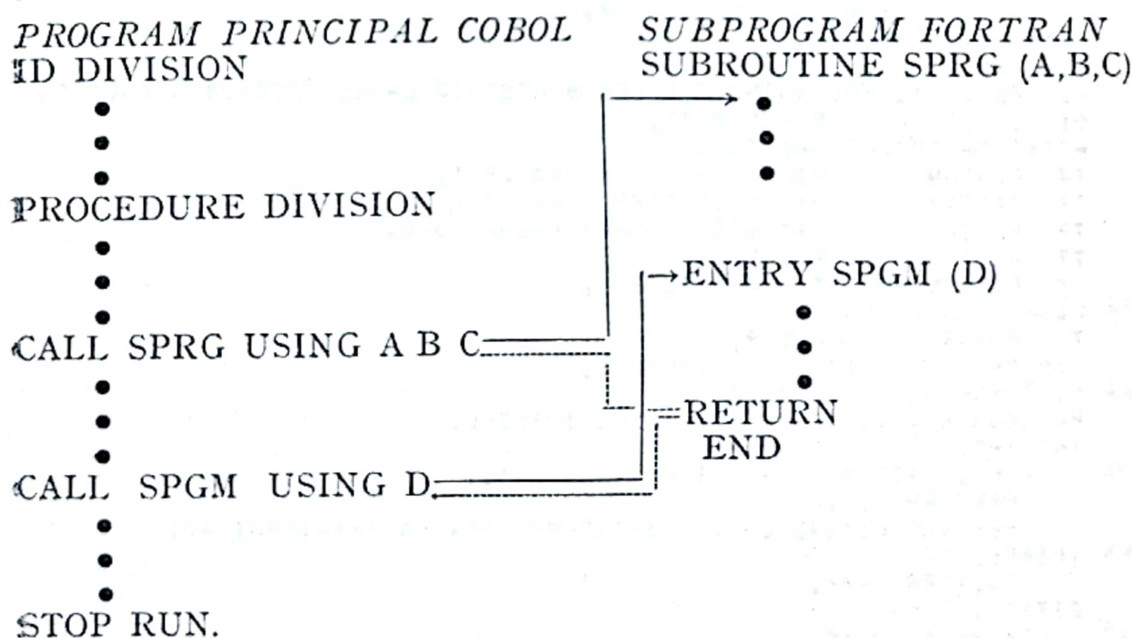
R. Da, deși nu e uzual. În acest caz nu există transfer de date între program și subprogram. În subprogram dispare necesitatea lui USING în dreptul lui PROCEDURE DIVISION; de asemenea poate să dispară secțiunea LINKAGE.

20.9 Numele subprogramului FORTRAN * este fie cel asociat cu un enunț SUBROUTINE fie cel dintr-un ENTRY. Parametrii, dacă există, însoțesc în paranteză aceste nume.

Intrarea într-un subprogram FORTRAN este prima instrucțiune executabilă după SUBROUTINE respectiv după ENTRY.

Ieșirea se face prin instrucțiunea RETURN.

Schematic apelarea unui subprogram FORTRAN se face în felul următor:



* Se presupune în acest paragraf că cititorul are noțiuni elementare de FORTRAN.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRGM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  REVENIRE        PIC 9 VALUE 0.
01  INREG.
 2  MARCA.
  3  UTIL           PIC 9(5).
  3  CIF-CONT       PIC 9.
 2  NUMELE          PIC X(30).
 2  FUNCTIA         PIC X(10).
 2  LOC-MUNCA       PIC 99.
 2  FILLER          PIC X(200).
PROCEDURE DIVISION.
1.  PERFORM APELARE.
    PERFORM PRELUCRARE UNTIL REVENIRE = 1
    STOP RUN.
PRELUCRARE.
*       ... DIFERITE ENUNTURI DE PRELUCRARE
    PERFORM APELARE.
APELARE.
    CALL SECVENTA USING REVENIRE INREG.
```

Programul 20.1. Program principal COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SECVENTA.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT PERSONAL ASSIGN ADK
            ORGANIZATION DIRECT
            ACCESS RANDOM
            ACTUAL KEY CASETA
            SYMBOLIC KEY CONTOR.
DATA DIVISION.
FILE SECTION.
FD  PERSONAL RECORDING F BLOCK 8 RECORDS LABEL RECORDS STANDARD.
01  INREG        PIC X(248).
WORKING-STORAGE SECTION.
77  CONTOR       PIC 9(5)        VALUE 1.
77  CASETA       PIC 9(9) COMP   VALUE 0.
77  MAXIM        PIC 9(5) COMP-3 VALUE 3500.
77  WCIT         PIC 9.
 88 CITIRE-REUSITA       VALUE 1.
LINKAGE SECTION.
77  RETUR        PIC 9.
 88 S-A-TERMINAT         VALUE 1.
01  INREG-L      PIC X(248).
PROCEDURE DIVISION USING RETUR INREG-L.
INTRARE.
    IF CONTOR = 0 OPEN INPUT PERSONAL.
    PERFORM CITIRE
    PERFORM CITIRE UNTIL CITIRE-REUSITA OR S-A-TERMINAT.
IESIRE.
    EXIT PROGRAM.
CITIRE.
    MOVE 1 TO WCIT
    DIVIDE CONTOR BY 8 GIVING CASETA
    READ PERSONAL INVALID KEY MOVE 0 TO WCIT.
    MOVE INREG TO INREG-L.
    ADD 1 TO CONTOR
    IF CONTOR GREATER MAXIM MOVE 1 TO RETUR
       CLOSE PERSONAL.
```

Program 20.1. (continuare 1): Subprogram COBOL

```
 .          LINK
LINK        STARTED


LINK       15.06.06     14/01/80     10H45M22S


 SEGMENT     PRGMXX00    NO     1.     IMPLANTATION       0
             MODULE    PRGMXX01        IMPLANTATION      10
             MODULE    STOPRUN         IMPLANTATION      80
                                             LONGUEUR DU SEGMENT    F0
 SEGMENT     PRGMXX00    NO     2      IMPLANTATION      F0
             MODULE    PRGMXX00        IMPLANTATION       0
                                             LONGUEUR DU SEGMENT   158
 SEGMENT     SECVEX00    NO     3      IMPLANTATION     248
             MODULE    SECVEN01        IMPLANTATION      10
             MODULE    OPCLTYP         IMPLANTATION     198
             MODULE    CXDD84          IMPLANTATION     2A8
                                             LONGUEUR DU SEGMENT   360
 SEGMENT     SECVEX00    NO     4      IMPLANTATION     5A8
             MODULE    SECVEN00        IMPLANTATION       0
                                             LONGUEUR DU SEGMENT   208
 SEGMENT     SECVENA1    NO     5      IMPLANTATION     780
             MODULE    SECVENA1        IMPLANTATION       0
                                             LONGUEUR DU SEGMENT   800


LINK       15.06.06     14/01/80     10H45M23S

                        IMPLANT. APRES TRAITEMENT OPTION FMS

 SEGMENT     PRGMXX00    NO     1      IMPLANTATION       0
                                             LONGUEUR DU SEGMENT   500

 SEGMENT     PRGMXX00    NO     2      IMPLANTATION     500
                                             LONGUEUR DU SEGMENT   158

 SEGMENT     SECVEX00    NO     3      IMPLANTATION     728
                                             LONGUEUR DU SEGMENT   800

 SEGMENT     SECVEX00    NO     4      IMPLANTATION     FE8
                                             LONGUEUR DU SEGMENT   240

 SEGMENT     SECVENA1    NO     5      IMPLANTATION    1228
                                             LONGUEUR DU SEGMENT   800
```

Programul 20.1. (continuare 2). Editarea legăturor

Vezi și programul 20.2 din prezenta lecție.

```
 COMPILE COBOL                                                    COBOL. ANS.. 2.
STARTED

                 IDENTIFICATION DIVISION.
                 PROGRAM-ID. CALCUL.
                 ENVIRONMENT DIVISION.
                *    ...
                 DATA DIVISION.
                *    ...
                 WORKING-STORAGE SECTION.
                *    ...
                 77  X              COMP-1.
                 77  CONTOR         PIC 9(5) COMP-3 VALUE 0.
                 77  CONTED         PIC Z(5).
                 77  PERIOADA       PIC 9(5) COMP-3 VALUE 28.
                 77  CIT            PIC 9(5) COMP-3 VALUE 0.
                 77  REST           PIC 9(5) COMP-3 VALUE 0.
                 77  SINUSW         PIC S9V999 COMP-3.
                 77  EDSIN          PIC ---.999B.
                 77  I              PIC 99    COMP.
                 01  GRAFIC.
                  2  PUNCT OCCURS 21 PIC X.
                 PROCEDURE DIVISION.
                 P.  PERFORM REPREZINTA UNTIL CONTOR > 365.
                *    ...
                     STOP RUN.
                 REPREZINTA.
                     ADD 1 TO CONTOR
                     DIVIDE CONTOR BY PERIOADA GIVING CIT REMAINDER REST
                     DIVIDE REST BY PERIOADA GIVING X
                     CALL SINUSF USING X
                     MOVE X TO SINUSW EDSIN
                     MOVE CONTOR TO CONTED
                     ADD 1 TO SINUSW
                     MULTIPLY 10 BY SINUSW GIVING I
                     ADD 1 TO I
                     MOVE SPACES TO GRAFIC
                     MOVE '*' TO PUNCT (I)
                     DISPLAY CONTED EDSIN GRAFIC.


       COMPILE FORTRAN
*FORTRAN  STARTED
                                                     AIMOD02     14/01/80   10.54
FORTRAN 00.00

                          SUBROUTINE SINUSF (X)
                          X=X*6.28319
                          X=SIN(X)
                          RETURN
                          END
```

Programul 20.2. Program principal COBOL și subprogram FORTRAN

```
•        LINK
LINK     STARTED



LINK     15,06,06   14/01/80   10H42M32S


  SEGMENT   TEST2X00   NO   1    IMPLANTATION     0
            MODULE  TEST2X01     IMPLANTATION    38
            MODULE  TDESYSIN     IMPLANTATION   1C8
            MODULE  TDESYSUT     IMPLANTATION   240
            MODULE  OPCLTYP      IMPLANTATION   280
            MODULE  C76          IMPLANTATION   3C0
            MODULE  STOPRUN      IMPLANTATION   480
            MODULE  UTWRTPCH     IMPLANTATION   520
            MODULE  READ         IMPLANTATION   5A8
            MODULE  C75A         IMPLANTATION   678
            MODULE  CXWRITE2     IMPLANTATION   750
            MODULE  CXWRITEA     IMPLANTATION   7A8
            MODULE  CXWRITEB     IMPLANTATION   7E0
                                       LONGUEUR DU SEGMENT  838

  SEGMENT   TEST2X00   NO   2    IMPLANTATION   838
            MODULE  TEST2X00     IMPLANTATION     0
                                       LONGUEUR DU SEGMENT  1F8

  SEGMENT   TEST2X82   NO   3    IMPLANTATION   A30
            MODULE  TEST2X82     IMPLANTATION     0
                                       LONGUEUR DU SEGMENT  2E0

     RUN
   STARTED

  1     ,222                  *
  2     ,433                    *
  3     ,623                      *
  4     ,781                       *
  5     ,900                         *
  6     ,974                         *
  7    1,000                          *
  8     ,974                         *
  9     ,900                         *
 10     ,781                       *
 11     ,623                      *
 12     ,433                    *
355   -,900  *
356   -,974 *
357  -1,000 *
358   -,974 *
359   -,900  *
360   -,781   *
361   -,623    *
362   -,433      *
363   -,222         *
364    ,000            *
365    ,222              *
366    ,433                *
```

Programul 20.2. (continuare 1) Fragmente din datele listate în execuţie

Corespondența descrierii parametrilor este:

FORTRAN	*COBOL*
Intregi	PIC 9(9) COMP
Reali	COMP—1
Dublă precizie	COMP—2
Vector n elem. alfanumerice	PIC X(m) ;m = 4n

20.10 *Î.* Un cîmp descris cu PIC 9(5) COMP poate servi drept parametru cu un subprogram FORTRAN?

R. Da, deoarece PIC 9(5) COMP și PIC 9(9) COMP sînt echivalente, ambele generînd un cîmp binar de 4 octeți cadrat la limită de cuvînt. Parametrul FORTRAN va fi un număr întreg.

20.11 Numele apelat prin CALL trebuie să apară ca definiție externă în subprogramul ASSIRIS* (etichetă de CSECT sau operand într-o directivă DEF). Un același subprogram poate avea mai multe puncte de intrare.

Obligatoriu numele apelat trebuie să fie menționat într-o cartelă ENTSEG, referitor la segmentul în care se află subprogramul.

Directiva END a subprogramului ASSIRIS nu trebuie să specifice adresa de lansare. Revenirea la programul COBOL se face printr-o instrucțiune BRU 0,8 sau BRU*32. Dacă se face apel din subprogram la un alt subprogram sau se utilizează macroinstrucțiuni SGF este necesară ocrotirea (salvarea) registrului 8.

Dacă subprogramul utilizează registrele 9—13 acestea trebuie ocrotite și restaurate deoarece sînt folosite de programul COBOL.

Parametrii din USING se transmit printr-o listă formată din mai multe elemente avînd dimensiunea unui cuvînt. Adresa listei de parametri este plasată în registrul 2. În fiecare cuvînt din listă se plasează adresa unuia dintre operanzi și eventual lungimea, după regula:

— bit 0—3 zero binar
— bit 4—7 lungimea parametrului, dacă este sub 16 octeți
— bit 8—31 adresa absolută a parametrului, cadrată la dreapta.

Lista este precedată de un cuvînt conținînd numărul de parametri. Utilizarea acestui cuvînt permite transmiterea unui număr variabil de parametri.

Observație

Subprograme COBOL pot fi apelate de asemenea din programe ASSIRIS sau FORTRAN.

Legătura între limbajele COBOL, FORTRAN și ASSIRIS se poate realiza de asemenea, cu respectarea anumitor precauții, prin intermediul fișierelor.

20.12 Într-o anumită măsură modularitate există în orice program ea nu este însă exploatată totdeauna în mod consecvent. Într-o aplicație se disting de regulă următoarele nivele de modularitate:

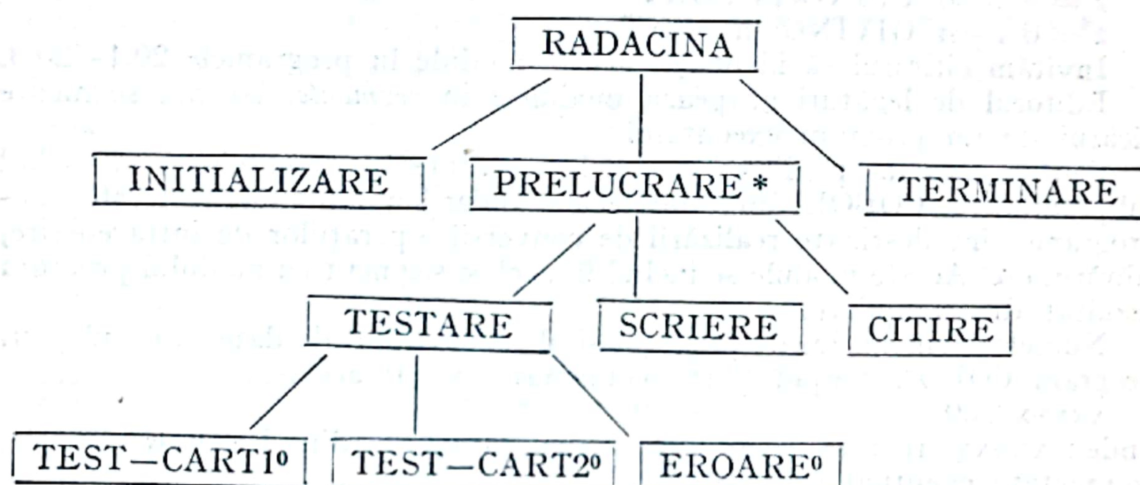
— subsistem
— flux

* Se presupune în acest paragraf că cititorul are noțiuni elementare de limbaj ASSIRIS.

— program
— modul de program
— secţiune/subprogram

Este foarte important ca aceste nivele să apară în mod clar în documentaţia aferentă. De asemenea alcătuirea fluxurilor să nu constituie obiectul unor improvizaţii.

20.13 La nivelul unui program structura modulară se poate reprezenta printr-o diagramă de ierarhie, indicînd succesiunea apelărilor. Dăm mai jos această diagramă pentru exemplul 20.3. din cadrul acestei lecţii. Observăm că prin „*" s-a notat o structură cu iteraţie iar prin „⁰" o structură de selecţie (condiţionată).



20.14 *Î.* Subprogramele TESTARE şi SCRIERE ar putea fi declarate şi ca subprograme separate apelate prin CALL. În ce caz preferăm ca un subprogram să fie extern, apelat prin CALL? Enumeraţi cîteva motive.

R. Cele mai importante motive ar putea fi:
— Dacă, formînd un singur program, dimensiunile acestuia ar fi prea mari.
— Dacă subprogramul ar putea fi apelat şi de către alte programe
— Dacă subprogramul este scris în alt limbaj decît COBOL.

20.15 Termenul *modul* se foloseşte (într-un sens mai restrîns) pentru a numi cea mai mică unitate tratată ca entitate separată de către editorul de legături.

În acest sens compilatorul produce dintr-un program sursă COBOL simplu (fără segmentare) următoarele:

— un modul al instrucţiunilor de program (P)
— un modul de date (D)
— module „comune" (C)

Modulele comune se generează cîte unul pentru fiecare fişier cu idex, reprezentînd ariile de intrare/ieşire aferente.

Numele modulelor COBOL se formează în felul următor:
xxxxxx01 pentru modulul *program*
xxxxxx00 pentru modulul de *date*
xxxxxxyz pentru un modul „*comun*"
unde xxxxxx reprezintă primele şase caractere din numele (din PROGRAM—ID al) programului, completat eventual cu caractere procent (%), şi y este idexul iar z tipul fişierului, adică
z = 1 fişier deschis INPUT
z = 2 fişier deschis OUTPUT
z = 3 fişier deschis INPUT—OUTPUT
z = 4 fişier deschis REVERSED
z = 5 fişier USING în SORT
z = 6 fişier GIVING în SORT
Invităm cititorul să identifice aceste module în programele 20.1—20.3.

20.16 Editorul de legături grupează modulele în *segmente*, iar din segmente alcătuieşte un program executabil.

Pentru aceasta, pe lîngă modulele rezultate din compilare, se includ subprogramele COBOL din biblioteca standard a sistemului. Aceste subprograme sînt destinate realizării de conversii, operaţiilor de intrare/ieşire, editării, etc. Aceste module se includ în acelaşi segment cu modulul program rezultat din compilare.

Numele segmentului de program şi al segmentului de date în cazul unui program COBOL simplu (fără segmentare) va fi acelaşi:
xxxxx%00
unde: xxxxx reprezintă primele cinci caractere din PROGRAM—ID, completate eventual cu procente.

Segmentul de date ia naştere din modulul de date. Din fiecare modul „comun" ia naştere cîte un segment „comun" avînd acelaşi nume ca şi modulul.

20.17 *Î.* Examinaţi exemplul 20.1 din prezenta lecţie.
1. Programul . . . . . . apelează subprogramul . . . . utilizînd parametri numiţi în programul principal prin . . . . .
2. Editorul de legături alcătuieşte . . . segmente, dintre care . . din programul principal şi . . . . .din subprogram.
3. Primul segment este format din . . module
4. Există un singur segment „comun" numit . . . . . şi destinat citirii fişierului . . . . în programul . . . .

R. 1. PRGM, SECVENTA, REVENIRE, INREG
2. 5, 2, 3
3. 2
4. SECVENA1, PERSONAL, SECVENTA

20.18 În condiţiile în care memoria operativă la dispoziţia unui program e limitată, structura arborescentă a programelor (vezi diagrama de structură de la 20.13) poate sugera un mod de lucru în care la un moment dat numai o parte dintre segmentele programului se află în memoria operativă, aducerea segmentelor în memorie făcîndu-se conform cerinţelor logicii

```
                IDENTIFICATION DIVISION.
                PROGRAM-ID. RADACINA.
                ENVIRONMENT DIVISION.
                INPUT-OUTPUT SECTION.
                FILE-CONTROL.
                    SELECT FISCART ASSIGN TO SYSIN.
                    SELECT FISDISC ASSIGN TO ADK.
                DATA DIVISION.
                FILE SECTION.
                FD  FISCART LABEL RECORDS OMITTED RECORDING F.
                01  CART1.
                 2  TIP           PIC X.
                 2  A             PIC X(9).
                 2  B             PIC 9(7).
                 2  C             PIC X(20).
                01  CART2.
                 2  FILLER        PIC X.
                 2  D             PIC X(30).
                 2  E              IC 9(4)V99.
                FD  FISDISC LABEL RECORDS STANDARD RECORDING F.
                01  INREGISTRARE.
                 2  F             PIC X.
                2   G             PIC X(30).
                 2  H             PIC 9(7)V99.
                WORKING-STORAGE SECTION.
                77  WSF           PIC 9 VALUE 0.
                 88 SFIRSIT             VALUE 1.
               *    ...
                PROCEDURE DIVISION.
                PRINCIPAL SECTION.
                RADACINA.
                    PERFORM INITIALIZARE.
                    PERFORM PRELUCRARE UNTIL SFIRSIT.
                    PERFORM TERMINARE.
                INITIALIZARE.
                    OPEN INPUT FISCART OUTPUT FISDISC.
               *    ...
                    PERFORM CITIRE.
                TERMINARE.
                    CLOSE FISCART FISDISC.
               *    ...
                    STOP RUN.
                PRELUCRARE.
                    PERFORM TESTARE
                    PERFORM SCRIERE
                    PERFORM CITIRE.
```

COBOL*ANS** 2, ℓ

```
                    CITIRE.
                        READ FISCART AT END MOVE 1 TO WSF.
                   *    ...
                    TEST SECTION 50.
                    TESTARE.
                        IF TIP = '1' PERFORM TEST-CART1 ELSE
                        IF TIP = '2' PERFORM TEST-CART2 ELSE
                                     PERFORM EROARE.
                    TEST-CART1.
                   *    ...
                    TEST-CART2.
                   *    ...
                    EROARE.
                   *    ...
                    SCRIERE SECTION 60.
                    SCR.
                   *    ...
                        WRITE INREGISTRARE.
```

Programul 20.3

```
 1          LINK
LINK       STARTED


LINK      15,06,06    14/01/80   10H34M37S


  SEGMENT     RADAC%00   NO    1     IMPLANTATION      0
              MODULE   RADACI01      IMPLANTATION     20
              MODULE   STOPRUN       IMPLANTATION    130
              MODULE   TDFSYSIN      IMPLANTATION    1A0
              MODULE   OPCLTYP       IMPLANTATION    218
              MODULE   READ          IMPLANTATION    328
                                              LONGUEUR DU SEGMENT   3F8

  SEGMENT     RADAC%00   NO    2     IMPLANTATION    3F8
              MODULE   RADACI00      IMPLANTATION      0
                                              LONGUEUR DU SEGMENT   150

     GMENT    RADACIA2   NO    3     IMPLANTATION    548
              MODULE   RADACIA2      IMPLANTATION      0
                                              LONGUEUR DU SEGMENT    6P

  SEGMENT     RADAC%01   NO    4     IMPLANTATION    5B0
              MODULE   RADACI02      IMPLANTATION      8
                                              LONGUEUR DU SEGMENT    70

  SEGMENT     RADAC%11   NO    5     IMPLANTATION    5B0
              MODULE   RADACI12      IMPLANTATION      8
              MODULE   UTWRTPCH      IMPLANTATION     30
                                              LONGUEUR DU SEGMENT    C0



LINK      15,06,06    14/01/80   10H34M38S

                         IMPLANT, APRES TRAITEMENT OPTION FMS

  SEGMENT     RADAC%00   NO    1     IMPLANTATION      0
                                              LONGUEUR DU SEGMENT 16E0

  SEGMENT     RADAC%00   NO    2     IMPLANTATION   16E0
                                              LONGUEUR DU SEGMENT  1A0

  SEGMENT     RADACIA2   NO    3     IMPLANTATION   1880
                                              LONGUEUR DU SEGMENT   68

  SEGMENT     RADAC%01   NO    4     IMPLANTATION   18E8
                                              LONGUEUR DU SEGMENT   70

  SEGMENT     RADAC%11,  NO    5     IMPLANTATION   18E8
                                              LONGUEUR DU SEGMENT   C0
```

…ntinuare 1) Editarea legătur…

prelucrării .Un astfel de mod de lucru se numește *„cu suprapunere”* sau *„cu acoperire”*.

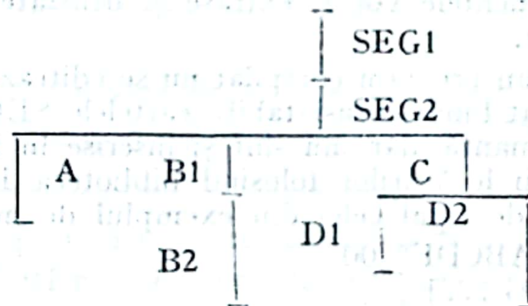
Pentru programul 20.3 subprogramele TEST și SCRIERE ar putea avea alocat același loc în memorie. În timpul testării cartelei în memoria operativă se va afla subprogramul TEST iar în timpul scrierii subprogramul SCRIERE, ș.a.m.d. alternativ. Bineînțeles aducerea subprogramelor în memorie se va face cu un consum suplimentar de timp pentru rulare.

20.19 Presupunem că un program (inclusiv subprogramele sale) este alcătuit din segmentele notate SEG1, SEG2, A, B1, B2, C, D1, D2. În diferite momente ale prelucrării este avantajos să se afle simultan în memorie următoarele grupuri de segmente

SEG1, SEG2 și A
SEG1, SEG2, B1 și B2
SEG1, SEG2, C și D1
SEG1, SEG2, C și D2

Segmentele unui program prezente simultan în memoria operativă (la un moment dat) poartă numele de *ramură*. Pentru programul de mai sus avem deci patru ramuri.

Se observă că ramurile au o parte comună. Ea poartă numele de *rădăcină* și trebuie să se afle în permanență în memorie. În esemplul nostru rădăcina este formată din segmentele SEG1, și SEG2. Arborescența segmentelor se poate reprezenta ca în figura ălăturată.



Apelul unui program prin CALL nu este posibil decît între segmente din aceeași ramură.

O structură de program cu suprapunere rezuîtă numai dacă se cere aceasta, explicit sau implicit. Altminteri segmentele programului vor forma o singură ramură, deci o structură fără suprapuneri.

Calea cea mai generăîă pentru a obține o structură cu suprăpunere este utilizarea cartelei TREE*.

20.20 *Cartela TREE* este o cartelă de comandă și are punct în col. 1 și „TREE” începînd din coloana 11. În continuare se specifică segmentele care formează arborescența, avînd între ele semnele +, ( ) cu următoarele semnificații:

\+ prelungirea unui segment cu un segment „comun”
(trecerea la un nivel inferior al arborescenței
, trecerea la o ramură paralelă, la același nivel

* TREE — arbore

) revenirea la nivelul superior;
numărul parantezelor stîngi şi drepte trebuie să fie egale.

Exemplu:
Pentru arborescenţa din 20.19 cartela de TREE va fi:
TREE SEG1+SEG2(A,B1(B2),C(D1,D2))
sau
TREE SEG1(SEG2(A,B1(B2),C(D1,D2)))
Cartela TREE trebuie plasată imediat înainte de cartela LINK.

20.21 *Î.* 1. De ce nu este corectă cartela următoare
TREE X(A(M(Y,R),C(I))?
2. În arborescenţa din 20.19 se fac următoarele apelări prin CALL: SEG2 din B2, B2 din B1, A din C, D1 din C, B2 din A, D2 din SEG2, SEG2 din D1. Care dintre ele sînt incorecte?

R. 1. îi lipseşte o paranteză dreaptă (închisă)
2. A din C, B2 din A.

20.22 Cartela TREE trebuie precedată de cartelele:
SEG identificarea unui segment
ENTSG identificarea unei intrări în segment.

Compilatorul COBOL prelucrînd un program sursă produce automat şi cartelele SEG, ENTSG şi TREE (dacă este cazul), plasîndu-le, împreună cu modulele obiect, în fişierele de sistem FILEDIT şi REPEDIT. Din aceste fişiere componentele vor fi extrase şi utilizate pentru operaţia de editare a legăturilor.

În cazul în care un program compilat nu se editează ci se depune într-o bibliotecă BT (format binar translatabil), cartelele SEG, ENTSG şi TREE sînt listate la imprimantă dar nu sînt şi înscrise în bibliotecă. De aceea, în momentul editării legăturilor folosind biblioteca BT va fi necesară o secvenţă de cartele de tipul celei din exemplul de mai jos:

```
  .    SEG ABCDE%00
[ .    ENTSG ...]
  .    FETCHB FN: ABCDEF00,LN ...
  .    SEG ABCDE%01
[ .    ENTSG ...]
  .    FETCHB FN: ABCDEF01,LN...
  .    SEG ABCDE%02
[ .    ENTSG...]
  .    FETCHB FN: ABCDEF02,LN ...
       TREE ABCDE%00 (ABCDE%01,ABCDE%02)
```

Remarcă
Segmentele de date sînt implicite şi nu trebuie specificate. Însă dacă dorim ca un segment comun să nu aparţină rădăcinii, el trebuie citat explicit.

20.23 Un program COBOL poate chema un subprogram sau mai multe scrise în COBOL sau într-un alt limbaj. Subprogramele COBOL vor constitui de regulă segmente separate. Subprogramele ASSIRIS sau FORTRAN pot forma segmente independente sau pot fi încorporate segmentelor COBOL.

Exemplu

În exemplul 20.1 din prezenta lecţie programul principal COBOL, ,,PRGM'' apelează subprogramul COBOL ,,SECVENTA''. Cele două programe se compilează succesiv, după care se realizează editarea legăturilor care dă naştere la cinci segmente:

PRGM%%00 — segment program din PRGM
PRGM%%00 — segment de date din PRGM
SECVE%00 — segment program din SECVENTA
SECVE%00 — segment de date din SECVENTA
SECVE%A1 — segment ,,comun'' din SECVENTA pentru fişierul de idex A1.

Se observă că cele cinci segmente sînt implantate unul după celălalt, formînd o singură ramură (fără suprapuneri).

20.24 *Î.* Examinaţi programul 20.2 în care un program principal COBOL numit CALCUL apelează un subprogram FORTRAN numit SINUSF, precum şi mesajele editorului de legături.

1. Cîte segmente alcătuieşte editorul de legături şi care este tipul lor?
2. Cum se numesc modulele de program ale celor două programe şi în ce segmente sînt plasate?

R. 1. două, primul de program, al doilea segmentul de date (al programului COBOL)
2. CALCUL01, SINUSF, ambele plasate în segmentul CALCU%00

Remarcă

În segmentul CALCU%00 se găsesc de asemenea subprograme din biblioteca COBOL (de ex. STOPRUN pentru terminare, C%DDFI de conversie în virgulă mobilă, C76 pentru DISPLAY) şi din biblioteca FORTRAN (de ex. F%ASIN pentru funcţia sinus).

20.25 O structură cu suprapunere se realizează ca urmare a unei cereri explicite într-o cartelă TREE, dar compilatorul COBOL poate realiza şi el structuri cu suprapunere (generînd între altele automat şi cartelaTREE corespunzătoare) în cazul utilizării segmentării COBOL.

În acest caz diviziunea PROCEDURE este împărţită pe secţiuni, fiecare secţiune avînd un număr de prioritate.

Secţiunile sînt de două feluri, după cum vor forma:

— partea fixă (rădăcina arborescenţei)
— segmentele independente (ramurile care se suprapun).

Clasificarea segmentelor se face alocîndu-se fiecărei secţiuni o prioritate, după formatul:

nume-secţiune *SECTION* [număr prioritate].

Numărul de prioritate se poate acorda între 0 şi 99. Dacă lipseşte se consideră egal cu zero.

Secţiunile cu prioritate 0—49 formează un singur segment — partea fixă.

Secţiunile cu prioritate 50—99 devin segmente independente.

Pentru secțiunile DECLARATIVES nu este permisă alocarea de priorități, acestea fiind (deci) de prioritate zero.

Numele modulelor generate va fi:

xxxxxx01 modul rădăcină
xxxxxx00 modul date
xxxxxxuu modul ramură
xxxxxxyz modul comun

unde uu = număr prioritate — 48 (Pentru xxxxxx și yz vezi 20.15).

Numele segmentelor este același ca și în cazul programelor nesegmentate (vezi 20.15). În plus, pentru segmentele ramură se atribuie numele:

xxxxx%vv

unde xxxxx reprezintă primele 5 caractere din PROGRAM—ID, completate eventual cu % iar

vv = nume-prioritate — 49

Structura de arborescență constă dintr-o rădăcină și ramuri puse toate în paralel. Dacă se dovedește schimbarea acestei tructuri de arborescență se va include înainte de LINK o cartelă TREE cu arborescența dorită. Această cartelă va prevala asupra enunțului TREE generat de compilator.

20.26 *Î*. Examinați programul 20.3 din prezenta lecție.

1. Diviziunea PROCEDURE conține . . . . secțiuni
Prioritățile acestora sînt, în ordine . . . .
2. Se generează . . . . segmente. Dintre acestea . . . segmente sînt independente (ramură). Numele lor este . . . . ele sînt implantate începînd cu adresa relativă . . .
3. S-ar putea face un apel prin PERFORM dintr-o secțiune în alta. Care dintre apelurile de mai jos sînt incorecte? De ce?
din RADACINA în TEST
din TEST în RADACINA
din SCRIERE în TEST
din SCRIERE în RADACINA

R. 1. 3, 0, 50, 60
2. 5, 2, RADAC%01, RADAC%11, 18E8
3. Din SCRIERE în TEST. Cele două secțiuni nu se află nici dată simultan în memorie.

LECŢIA a 21-a TIPURI DE ERORI ŞI TEHNICI DE DEPANARE A PROGRAMELOR

- erori de compilare
- erori de editare a legăturilor
- erori de lansare şi execuţie
- depanarea programelor
- tehnici de testare

21.1 Erorile într-un program COBOL sînt uneori clasificate în:
— erori sintactice (de formă)
— erori logice (de conţinut)
Erorile sintactice sînt cele semnalate de compilator, editorul de legături sau la tratarea cartelelor de comandă. Erorile logice sînt cele care apar în cursul execuţiei. În general erorile sintactice sînt mult mai uşor de reperat şi înlăturat decît cele logice.

21.2 *Î.* *Erorile de compilare* au fost prezentate în lecţia 16.
1. Nivelele de gravitate ale erorilor de compilare sînt . . . .
2. Nivelul maxim de eroare admis se specifică prin opţiunea . . . din cartela . . . . În caz că lipseşte această opţiune nivelul se ia egal cu . . . .

R. 1. 0, 1, 2, 3, 4,
2. ER : n, COMPILE, n = 2

În cazul în care există cel puţin o eroare de nivel mai mare decît n şi se cere editarea legăturilor, se generează mesajul

EDITION DES LIENS IMPOSSIBLE * ABANDON DU TRAVAIL

O listă a tuturor mesajelor de compilare se poate obţine specificînd opţiunea COBOL : DTB. Mesajele se pot obţine în limba franceză (DLG : FRA), română (DLG : ROM) sau germană (DLG : GER).

Opţiunea CNV va sugera schimbări în program care să ducă la reducerea numărului de conversii, mari consumatoare de timp.

21.3 *Î.* 1. Mesajele de nivel 0 se obţin doar în cazul utilizării multifişierelor sau dacă se specifică opţiunea . . . .
2. Dacă nu se specifică ER : n atunci erorile de nivel . . . interzic editarea legăturilor şi catalogarea programelor.

R. 1. CNV (conversii)
2. 3, 4

21.4 **Î.** Înlăturarea erorilor listate după o compilare nu este o garanție că la următoarea compilare nu vor apare alte mesaje de eroare. De ce?

R. Unele erori pot masca alte erori grave, în așa fel încît la prima compilare compilatorul să nu le mai poată remarca pe acestea din urmă. De asemenea un mesaj de nivel 4 poate lăsa porțiuni de program sursă neexaminate.

21.5 Editarea legăturilor se cere prin cartela LINK. Erorile apărute în cadrul acestei operații se referă de multe ori la sintaxa cartelelor de comandă și așa cum am văzut au 4 nivele de gravitate.

Fiecare mesaj este caracterizat de un cod de patru cifre din care prima cifră indică nivelul erorii (Anexa 2).

Generarea programului nu se produce dacă există cel puțin o eroare de nivel mai mare decît nivelul declarat prin opțiunea ER : n din cartela LINK. Prin omisiune se ia n = 1; nu se poate lua n = 4.

Dacă în cursul editării legăturilor s-a depășit nivelul permis de eroare, se generează mesajul:

EXECUTION IMPOSSIBLE * ABANDON DU TRAVAIL

și execuția lucrării se întrerupe.

21.6 În timpul încărcării programelor, operație comandată prin cartela RUN, se pot produce incidente care fac imposibilă rularea (Anexa 2).

Mesajele de eroare sînt însoțite de parametrii care oferă informații suplimentare pentru efectuarea corecțiilor.

21.7 **Î.** Erorile de editare a legăturilor și din timpul încărcării sînt erori sintactice sau logice?

R. Sintactice, deoarece se referă la respectarea cerințelor de formă ale programului și cartelelor de comandă.

21.8 În timpul execuției unui program (care a trecut cu succes de compilare, editare și lansare) pot apare:
- incidente detectate de module SGF
- erori fizice de intrare/ieșire
- erori de program
- întreruperea rulării la cererea operatorului.

21.9 Evenimentele detectate *de modulele SGF* duc la mesaje care se afișează la consolă. Dacă este vorba de un incident care provoacă oprirea anormală a programului mesajul apare și la imprimantă.

Formatul mesajului de consolă este diferit în funcție de evenimentul semnalat, ca de ex.:
- deschidere/închidere fișier
- creare punct de reluare
- schimbare volum
- alocare zonă de extensie
- blocuri scurte pe bandă (sub 12 octeți).

Mesajele de eroare și reluare la consolă au formatul:

NN AAA F XY OC NR TEXT

unde:

- *NN* — numărul partiției;
- *AAA* — adresa fizică a perifericului;
- *F* — litera 'F';
- *XY* — idexul complet (X — idex, Y — tip deschidere);
- *OC* — numărul segmentului OPEN/CLOSE corespunzător erorii;
- *NR* — numărul mesajului de eroare;
- *TEXT* — mesaj abreviat de eroare.

În caz de oprire anormală, mesajul corespunzător, listat la imprimantă, are formatul:

XY EGF OC NR TEXT

unde EGF reprezintă chiar literele 'EGF' iar celelalte componente au semnificația de mai înainte.

21.10 **Exemple** (Anexa 2):

Printre cele mai frecvente mesaje SGF amintim următoarele: tentativă de deschidere a unui fișier deja deschis; fișier inexistent; fișier existent deja; etc.

21.11 *Î.* În caz de incidente legate de prelucrarea fișierelor, modulele . . . afișează la . . . . mesaje de eroare. Aceste mesaje apar la . . . . doar în cazul în care incidentul duce la . . . .

Mesajele sînt (aceleași/diferite) pentru COBOL FORTRAN, ASSIRIS.

R. SGF, consolă și imprimantă, imprimantă, oprirea anormală a programului, aceleași.

21.12 *Erorile de intrare/ieșire* (vezi anexa 2) au formatul:

ERREUR E/S yyxxxxxx

unde:

- *yy* — numărul erorii
- *xxxxxx* — o adresă memorie, asociată de regulă blocului de control.

Exemplu: ERREUR E/S 20 ... reprezintă un incident ireparabil pe disc.

21.13 *Erorile de program* (vezi anexa 2) au formatul:

ERREUR PROGRAMME 0000xxxx

unde: xxxx este codul erorii

Mesajul este urmat de vidajul zonei de memorie corespunzătoare partiției.

Remarcă

1. Apariția mesajelor exemplificate mai sus într-un program COBOL are drept cea mai frecventă cauză utilizarea unor indici cu valoare zero sau superioară limitei maxime declarate prin OCCURS.

2. Mesajele eroare de program sînt independente de limbajul în care s-a scris programul în cauză. Printre mesaje există și unele privitoare la depășirea capacității în cazul operațiilor aritmetice. Pentru programe scrise în COBOL de regulă incidentele de acest tip sînt mascate deci nu se generează nici mesajele corespunzătoare, fiind de datoria programatorului să se asigure că operațiile aritmetice sînt corect executate de către calculator.

21.14 Întreruperea rulării la cererea operatorului provoacă unul dintre următoarele mesaje:

1. DECISION OPERATEUR xxxxxxxx

unde:

- *xxxxxxxx* conţine numărul erorii şi adresa blocului de control corespunzător urmat de vidajul memoriei. Oprirea execuţiei a fost cerută de operator în urma încercării nereuşite de a repara un incident apărut la o unitate periferică.

2. INTERVENTION OPERATOR, urmat de vidaj.

Oprirea s-a cerut prin butonul de întrerupere de la consolă, în cazul în care operatorul apreciază că programul lucrează anormal (de exemplu nu mai iese dintr-o ciclare).

21.15 *Î.* În cazul unei împărţiri cu zero într-un program COBOL apare mesajul . . . .

R. Nu apare nici un mesaj. În cîmpul destinat rezultatului apare un număr eronat şi rularea continuă. Înainte de o împărţire trebuie deci să ne asigurăm prin program că împărţitorul e diferit de zero.

21.16 *Î.* Într-un program COBOL avem:

```
77      I      PIC      9.
        . . .
        PERFORM S VARYING I FROM 1
        BY 1 UNTIL I>9
```

Care este eroarea? Ce mesaj se va imprima?

R. Ieşirea din ciclarea cu PERFORM nu se face decît dacă I devine egal cu 10, or, fiind declarat PIC 9, cîmpul I nu va putea fi niciodată egal cu 10. Rezultă că programul va cicla indefinit. Ciclarea ea însăşi nu produce nici un mesaj, programul poate fi însă oprit de operator (în acest caz apare mesajul INTERVENTION OPERATEUR) sau prin depăşirea timpului afectat prin cartela RUN.

21.17 Opţiunea SRG specificată în cartela COMPILE are ca efect generarea unor secvenţe de instrucţiuni pentru controlul valorii indicelui. În cazul în care un indice depăşeşte valorile limită ale tabloului pentru care se utilizează, la consolă va apare mesajul:

NN EN XXXXXX DEBORDEMENT DE L'INDICE: VVVVV

unde:

- *NN* — numărul partiţiei;
- *XXXXX* — adresa absolută de memorie;
- *VVVV* — valoarea indicelui.

Programul este oprit. La consolă se poate da unul dintre răspunsurile:

S(stop) execuţia este terminată

G(go) rularea continuă, în ciuda indicelui greşit.

21.18 Opţiunea SRG realizează şi un control relativ la fişiere, semnalînd la consolă erorile:

E/S SUR FICHIER NON OUVERT — operaţie de citire sau scriere pe fişier nedeschis;

OPEN SUR FICHIER OUVERT — tentativă de a deschide un fişier deja deschis;

CLOSE SUR FICHIER FERME — tentativă de a închide un fişier deja închis.

mesajul fiind precedat de cifrele 1, 2 sau 3 după tipul fişierului (INPUT, OUTPUT, I—0).

Ca şi la 21.17, se aşteaptă un răspuns la consolă:

- S(stop) execuţia este terminată
- G(go) eroarea este ignorată, prelucrarea continuă.

Opţiunea SRG este implicită dacă se specifică opţiunea DBG (vezi 21.19).

21.19 *Opţiunea DBG*

Dacă este prevăzută într-o cartelă COMPILE COBOL ea atrage după sine generarea unor instrucţiuni de depanare, scrise în programul sursă COBOL de către autorul programului; în plus opţiunea DBG va implica opţiunile CNV (vezi 21.2) şi SRG (vezi 21.15 şi 21.16).

Enunţurile de depanare pot fi prevăzute în oricare dintre cele patru diviziuni ale unui program COBOL şi pot conţine orice instrucţiune sau clauză permisă în contextul respectiv. Enunţurile de depanare vor avea semnul + perforat în coloana 7 a cartelei, respectiv semnul / pentru o cartelă de continuare.

Dacă în cartela COMPILE este prevăzută opţiunea DBG enunţurile de depanare se generează şi se execută ca şi toate celelalte. În cazul în care opţiunea DBG lipseşte, enunţurile de depanare sînt sărite de compilator, dar reproduse la imprimantă în chip de comentarii.

Avantajul utilizării metodei este că trecerea de la faza de depanare la cea de execuţie se poate face fără modificări în programul sursă doar printr-o simplă modificare în cartela COMPILE.

21.20 *Î.* 1. Opţiunile . . . . . din cartela . . . . servesc depanării programului în timpul rulării.

2. Opţiunea . . . implică opţiunile . . . . şi . . . . .

3. Opţiunea . . . . . . ne permite să realizăm modificări în program pentru a mări rapiditatea prelucrărilor.

4. Opţiunea . . . . elimină, prin controlul indicilor, majoritatea erorilor de tip . . . .

5. Opţiunea . . . . determină generarea unor enunţuri speciale pentru a controla mersul programului. Aceste enunţuri se scriu pe cartele avînd semnul . . sau . . . / în coloana . .

R. 1. DBG, SRG, COMPILE

2. DBG, SRG, CNV

3. CNV (vezi 21.2)

4. SRG (vezi 21.17., 21.18), ERREUR PROGRAMME (vezi 21.13)

5. DBG, +, /, 7

Enunțurile de depanare (vezi 21.19) vor urmări cel mai adesea furnizarea a două tipuri de informații:
— valorile intermediare ale unor variabile dacă se constată că valorile finale nu sînt cele scontate
— succesiunea executării paragrafelor, dacă există bănuiala că ea un se face în ordinea dorită.
În ambele cazuri se poate folosi instrucțiunea DISPLAY sau PRINT
Exemple
+ DISPLAY 'X =' X 'Y =' Y
+ DISPLAY 'A INCEPUT PARAGRAFUL Z'
+ PRINT W

21.21 *Î*. 1. Enunțurile de depanare folosesc cel mai adesea verbele . . . . sau . . . . Dintre acestea verbul . . . . duce la o mai mare economie în scriitura programului sursă.
2. Prin executarea unui PRINT se tipăresc la imprimantă . . . .

R. 1. PRINT, DISPLAY, PRINT
2. numele paragrafului, numele cîmpului/cîmpurilor de date și valoarea/valorile conținute, respectiv literalul/literalele specificate.

21.22 În cazul în care apar erori grave, rularea programului este terminată. În această situație sistemul de operare generează un *vidaj* al memoriei, adică o imagine completă, în hexazecimal (stînga) și caractere, (dreapta). a partiției afectate programului; prin acest serviciu programatorul poate obține o seamă de informații caracterizînd momentul întreruperii execuției.

21.23 Zona de memorie alocată unui program se împarte în modul următor:
— zona de informații generale (ZIG)
— tabelul de segmente (TS)
— zona program (ZP)
— zona de opțiuni (ZOP)

R14 conține adresa de început a segmentului de program în care s-a produs întreruperea

R13 conține adresa segmentului de date utilizat înainte de întrerupere;
R8 conține adresa de revenire în programul principal, în cazul apelării unui subprogram prin CALL.

21.24 *Zona de informații generale* (ZIG) conține un mare număr de adrese, contoare, indicatori și alte tipuri de informații. Între acestea menționăm:
1— 3 adresa modulului de legături SGF
4— 7 adresa rezumatului SGF
8—15 numele lucrării
18 codul de retur al fazei precedente (normal 0)
20—30 data calendaristică
32—35 adresa ultimului octet din partiție
36—39 adresa primului octet al zonei de opțiuni
40—43 adresa ultimului octet al ultimului segment încărcat în memorie
45—47 adresa ultimului octet al celei mai lungi ramuri.

21.25 *Rezumatul SGF* se află la sfîrşitul ZIG şi are structura :
— un cuvînt memorie la început, cu octeţii :
0, număr fişiere
1—3 adresa disc a fişierului de comunicaţii (FILECOM) în care se plasează de către monitor datele din cartelele ASSIGN, FILE, LABEL
— cîte 2 cuvinte de fiecare fişier, cu structura :
0 idex
1 tipul (intrare, ieşire, etc. vezi 20.15)
2 — 3 adresa relativă în FILECOM a fişierului
4 '00' — dacă fişierul nu e deschis
'01' — dacă fişierul e deschis
5 — 7 pentru fişierele deschise, adresa TDF*

21.26 *Tabela de segmente* (*TS*) se află plasat după ZIG şi are structura :
— un cuvînt memorie conţinînd adresa de început a partiţiei (primul octet al ZIG)
— cîte un cuvînt pentru fiecare segment, cu structura :
0 indicator :
'00' segment încărcat
'80' segment neîncărcat
(la segmente SORT regula e diferită)
1 — 3 adresa primului octet din segment

21.27 Examinaţi exemplele programul 21.1 şi 21.2 din cadrul acestei lecţii :
Î. 1. Adresele tabelelor de segmente pentru cele 2 programe sînt respectiv . . . . . şi . . . . . .
2. La programul 21.1 sînt . . . . . segmente încărcate la adresele . . . . . .
3. La programul 21.2 sînt . . . . segmente încărcate la adresele . . . . . .
4. La programul 21.2, în rezumatul SGF sînt reprezentate . . . . fişiere dintre care . . . . . deschise avînd adresele TDF . . . .
R. 1. 250BC, 250CC
2. 2, 250C8, 26558
3. 3, 250E0, 276B8, 278E0
4. 3, 3, 2, 252C4, 27820

21.28 *Geografia datelor* (MAP) permite localizarea în vidaj a cîmpurilor de memorie rezervate entităţilor descrise în diviziunea DATA, precum şi obţinerea altor informaţii asupra acestora.

Pentru datele din WORKING—STORAGE SECTION adresa din MAP (ADR) reprezintă deplasarea faţă de conţinutul registrului R13.

R13 + ADR

Pentru datele din FILE SECTION se adaugă şi argumentul ARCA dat pentru fiecare fişier în „geografia fişierelor" (GEOGRAPHIE DES FICHIERS)

R13 + ARCA + ADR

---

* detalii privind structura TDF se găsesc în NORMES DE PROGRAMATION — MANUEL D'UTILISATION

```
COMPILE COBOL OBL,MAP,CRF
STARTED
                                                        COBOL*ANS** 2,

       0001          IDENTIFICATION DIVISION,
       0002          PROGRAM-ID. TEST1,
       0003          ENVIRONMENT DIVISION,
     C 0004          *      ...
       0005          DATA DIVISION,
     C 0006          *      ...
       0007          WORKING-STORAGE SECTION,
       0008          77   I                     PIC 9(9).
     C 0009          *      ...
       0010          01   TABLOU.
       0011           2   ELEMENT   PIC X(20) OCCURS 100,
     C 0012          *      ...
       0013          PROCEDURE DIVISION,
     C 0014          *      ...
       0015          MUTARE.
       0016                 ACCEPT I
       0017                 MOVE '*' TO ELEMENT (I),
     C 0018          *      ...
       0019                 STOP RUN,
```

Programul 21.1. Program sursă COBOL

```
                                        GEOGRAPHIE DES. DONNEES

              NLSR    CAT      NIV    ADRESSE   LONGUEUR   TYPE   IDENTIFICATEUR

WORK SECTION

                 7              77     0032          2     BHW    TALLY
                 7              77     0034          2     BHW    RETURN-CODE
                 7              77     0038          4     BW     SORT-FILE-SIZE
                 7              77     003C          4     BW     SORT-CORE-SIZE
                 7              77     0040          4     BW     SORT-MODE-SIZE
                 7              77     0044          2     BHW    SORT-RETURN
                 8              77     0046          9     DE     I
                10              01     0050       2000     GR     TABLOU
                11              02     0050         20     AN     ELEMENT
```

Programul 21.1. (continuare 1) Geografia datelor

```
                                                                         COBOL*ANS** 2. 6*AITESTI *TESTI    *14/01/80*

ETAT OBL    *LTG.SOURC    *CODE VERBE *CPT.ORDI   *INSTRUCTION    *INSTRUCTION DECODEE                *SYMBOLICO
             0015                      0000        7E0A0000  X     LD4 ,14      15,0000
                                       0004        7D0A0000  X     LD4 ,13      15,0000
                                       0008        5F410000        LDTM,15      13,0000
          MUTARE                                                                  TEST0201
             0016         ACCEPT       000C        502B0046        LDDA, 0      13,0046           I
                                       0010        01240009        LD4I, 1        0009
                                       0014        012F0000  X     BSG , 1        0000            C%0087
             0017         MOVE         0018        012F0000  X     BSG , 1        0000            C%0091
                                       001C        50090046        DATA,4,4                       I
                                       0020        07290014        LD2I, 7        0014            I
                                       0024        07310018        MPU2, 7        0018
                                       0028        57AA003D        LD4I, 0      13,003D, 7        ELEMENT
                                       002C        012A0013        LD4I, 1        0013
                                       0030        00480040        CYBR, 0        0040
                                       0034        D7AB0874        LDDA, 0     *13,0874, 7        ELEMENT
                                       0038        51600870        ZAD , 1      13,0870
             0019         STOP         003C        012F0000  X     BSG , 1        0000            C%0061
          CXETFNPR                                                                  TEST0202
```

Programul 21.1. (continuare 2) Program obiect şi decodificarea în ASSIRIS

```
           LINK
LINK       STARTED


LINK       15,06,06    14/01/80    10H23M41S


  SEGMENT    TEST1%00   NO   1    IMPLANTATION      0
             MODULE   TEST1%01    IMPLANTATION     20
             MODULE   TOFSYSIN    IMPLANTATION     60
             MODULE   C87         IMPLANTATION     D8
             MODULE   C%DD82      IMPLANTATION    1A0
             MODULE   STOPRUN     IMPLANTATION    298
             MODULE   ACCEPT      IMPLANTATION    308
             MODULE   OPCLTYP     IMPLANTATION    348
                                       LONGUEUR DU SEGMENT   458

  SEGMENT    TEST1%00   NO   2    IMPLANTATION    458
             MODULE   TEST1%00    IMPLANTATION      0
                                       LONGUEUR DU SEGMENT   878
```

Programul 21.1. (continuare 3) Editare legături

```
 VIDAGE PARTITION 02

      ERREUR PROGRAMME 00000204

 ETAT PROGRAMME           00025118    F4140000

 REGISTRES          R0         R1         R2         R3         R4         R5         R6         R7
                 00122061   00000013   A1025144   00025160   00000012   00025520   C9FF0008   000FC7EC

                    R8         R9         R10        R11        R12        R13        R14        R15
                 00025104   00025000   0002AFC4   00025000   00000A60   00026558   000250C8   0002508C

 VIDAGE MEMOIRE      EV05

025000   00026078  00025080  C1C9E3C5  F2E3F140  025010  00FFFD00  F1F461F0  F161F8F0  F0F1F400   ---------AITEST1--
025020   0003DFFF  0003E000  000260CF  440260CF  025030  00000016  0000F000  00025046  0400040F   -----------------
025040   00025070  00000000  00000000  00000000  025050  00000000  00000000  00000000  00000000   -----------------
025060   00000000  00000000  00000000  00000000  025070  000250C8  00025400  A1025144  000251B0   -----------------
025080   00000012  00025520  000296C8  0002696A  025090  0002A64C  0000002D  C7D57A00  0002A510   -----------H-----
0250A0   00000000  000269BB  00025070  00026971  0250B0  01001E00  5CF100A0  01025144  00025000   -----------------
0250C0   000250C8  00026558  60000060  60000138  0250D0  60000200  600002B0  60000298  600002A4   ---H-------------
0250E0   60000306  60000428  7E0A0004  7D0A0008  0250F0  5F410000  502B0046  012A0009  E8390004   -----------------
025100   E8390008  50090048  07290014  07310018  025110  57AA003D  012A0013  004B0040  D7AB0674   Y----------------
025120   51600870  E639000C  7E0A0004  622A007C  025130  621B008B  650A009C  612A0084  08B80000   ----Y------------
025140   00010000  E25CF101  01000000  00000100  025150  00000080  00000050  00000000  00000000   ----S-1----------
      ZONE        0260E0       A       026120         IDENTIQUE
026120   052E0000  01010001  00026128  00026078  026130  00025474  00000000  00000012  00025520   -----------------
026140   052E0000  052E0000  052E0000  052E0000  026150  052E0000  052E0000  052E0000  052E0000   -----------------
      ZONE        026160       A       026540         IDENTIQUE
026540   052E0000  052E0000  052E0000  052E0000  026550  052E0000  052E0000  F00FFF50  00010000   -----------------
026560   00000000  00000000  C3D9C4C2  4B000000  026570  00000000  00000000  00000000  00000000   ---------CRDB-----
026580   00000000  00000000  00004040  00004040  026590  00000000  00000000  00000000  0000F9F9   -----------------
0265A0   F9F9F9F9  F9F9F900  00000000  00000000  0265B0  00000000  00000000  00000000  00000000   9999999----------
0265C0   00000000  00000000  00000000  00000000  0265D0  00000000  00000000  00000000  052E0000   -----------------
0265E0   052E0000  052E0000  052E0000  052E0000  0265F0  052E0000  052E0000  052E0000  052E0000   -----------------
      ZONE        026600       A       026DC0         IDENTIQUE
```

Programul 21.1. (continuare 4) Fragment din vidaj

```
COMPILE COBOL OBL,MAP,CRF
STARTED
                                                        COBOL*ANS** 2, 6*

IDENTIFICATION DIVISION.
PROGRAM-ID. TEST2.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FISCART ASSIGN TO SYSIN.
    SELECT FISBANDA ASSIGN TO BMT.
DATA DIVISION.
FILE SECTION.
FD  FISCART LABEL RECORDS OMITTED RECORDING F.
01  CARTELA.
  2 TIP-CART           PIC XXXX.
  2 COD-ARTICOL        PIC 9(8).
  2 CANTITATE          PIC 9(7).
  2 PRET-UNITAR        PIC 9(6)V99.
FD  FISBANDA LABEL RECORDS STANDARD RECORDING F
    BLOCK 20 RECORDS.
01  INREGISTRARE.
  2 B-COD              PIC 9(8).
  2 B-CANT             PIC 9(7) COMP-3.
  2 B-PRET             PIC 9(7)V99 COMP-3.
WORKING-STORAGE SECTION.
77  CONTOR-C           PIC 9(5) COMP-3 VALUE 0.
77  CONTOR-B           PIC 9(5) COMP-3 VALUE 0.
77  VALOARE-TOTALA     PIC 9(9)V99 COMP-3 VALUE 0.
77  VAL-EDITATA        PIC Z(9).99.
77  LUCRU              PIC 9(7)V99 COMP-3.
77  SF                 PIC 9 VALUE 0.
 88 SFIRSIT                  VALUE 1.
PROCEDURE DIVISION.
PRINCIPAL.
    OPEN INPUT FISCART OUTPUT FISBANDA.
    PERFORM CITIRE
    PERFORM CICLARE UNTIL SFIRSIT
    DISPLAY 'C:' CONTOR-C 'B:' CONTOR-B
    CLOSE FISCART FISBANDA.
    STOP RUN.
CICLARE.
    ADD 1 TO CONTOR-C
    MOVE COD-ARTICOL TO B-COD
    MOVE CANTITATE   TO B-CANT
    MOVE PRET-UNITAR TO B-PRET
    MULTIPLY PRET-UNITAR BY CANTITATE GIVING LUCRU
    ADD LUCRU TO VALOARE-TOTALA
    WRITE INREGISTRARE
    ADD 1 TO CONTOR-B
    WRITE CARTELA
    PERFORM CITIRE.
CITIRE.
    READ FISCART AT END MOVE 1 TO SF.
```

Programul 21.2

```
VIDAGE PARTITION 02

ETAT PROGRAMME                 00028474   F0140000

REGISTRES          R0         R1         R2         R3         R4         R5         R6         R7
                03027701   00025634   A202533B   000277A8   0000001B   00026470   0000001B   FFFFBEFA

                   R8         R9         R10        R11        R12        R13        R14        R15
                00025270   00025150   0002AFC4   FF02639E   00009A60   000276B8   000250E0   000250CC

VIDAGE MEMOIRE      EV05

025000   000271D8  00025080  C1C9E3C5  E2E3F240   025010   00FFA200  F1F461F0  F161F8F0  F0F1F400   ---Q----AITEST2-
025020   0003DFFF  0003E000  00027BBF  44027BBF   025030   0000001F  0000F000  00025048  0400040F   ----------------
025040   00025070  00000000  00000000  00000000   025050   00000000  00000000  00000000  00000000   ----------------
025060   00000000  00000000  00000000  00000000   025070   000250E0  00025634  A2027820  000277F6   ----------------
025080   00000011  80027840  000296C8  0002696A   025090   0002A64C  0000002D  C7D57A00  0002A510   -----------H----
0250A0   00000000  0002698B  00025070  00026971   0250B0   03001E00  5CF10088  010252C4  5CF24088   ----------------
0250C0   00000000  C2F28088  01027820  00025000   0250D0   000250E0  000276B8  000278E0  C3F14054   ----B2----------
0250E0   600001C8  60000240  60000280  60000390   0250F0   60000454  600004C8  60000490  600004BC   ---H------------
025100   60000520  600005A8  60000690  60000750   025110   600007A8  600007E0  7E0A0004  7D0A0008   ----------------
025120   5F410000  F8390000  032A0000  F8390008   025130   522A0168  525A0158  552A0188  555A015C   ----Y--------Y--
025140   550C015C  032A0000  F8390008  505C01F4   025150   603801A0  50180063  002400F1  6835008C   --------Y------4
025160   505C01F0  60380104  60380074  502801F8   025170   012A0002  E8390010  502801D8  012A0005   ---0-----------Y
025180   53670046  502801D8  012A0005  E8390010   025190   502801EA  012A0002  E8390010  502801DD   --------Q----Y--
0251A0   012A0005  53670049  502801DD  012A0005   0251B0   00280035  E8390010  032A0008  E8390000   ----------------
0251C0   F8390008  032A0008  550A015C  520A0158   0251D0   00290300  01290700  D3C40158  E8390008   Y---------------
0251E0   E8390014  532B0046  516101EC  58280140   0251F0   586000F4  502800FC  012A0007  54660148   Y---------------
025200   50280103  012A0008  5566014C  50280103   025210   012A0008  556601E1  502800FC  012A0007   ----------------
025220   596601D8  592801D8  556401E1  5528005E   025230   596001D8  5628004C  5561005E  042A0011   ---Q---Q--------
025240   520A0158  550A015C  532A0140  F8390020   025250   532B0049  516101EC  042A0018  532A00F0   -------------Y--
025260   E8390004  8D58000C  022800A2  88390014   025270   505C01F4  603801A0  590C01F0  14340004   Y---------------
025280   602801B4  532A00F0  F8390000  F8390024   025290   6038018C  002800F1  50580063  590C01F4   --------0Y---Y--
```

Programul 21.2 (cotinuarea 1)

```
              LINK
LINK        STARTED



LINK        15.06.06     14/01/80   10H46M53S


     SEGMENT     CALCU%00  NO    1      IMPLANTATION      0
                 MODULE   CALCUL01      IMPLANTATION     58
                 MODULE   SINUSF        IMPLANTATION    1D8
                 MODULE   TDFSYSUT      IMPLANTATION    220
                 MODULE   STOPRUN       IMPLANTATION    290
                 MODULE   C%DDFI8       IMPLANTATION    300
                 MODULE   C%FFIDE       IMPLANTATION    3C0
                 MODULE   CEDIT2        IMPLANTATION    458
                 MODULE   C%DDB4        IMPLANTATION    708
                 MODULE   C76           IMPLANTATION    7B8
                 MODULE   F%ASIN        IMPLANTATION    8A8
                 MODULE   C%DDB8        IMPLANTATION    A18
                 MODULE   C%BFI8        IMPLANTATION    AF0
                 MODULE   C%BBD8        IMPLANTATION    B50
                 MODULE   C%FIBI        IMPLANTATION    C48
                 MODULE   C75A          IMPLANTATION    D00
                 MODULE   C%WRITE2      IMPLANTATION    DD8
                 MODULE   F%ERR1        IMPLANTATION    E30
                 MODULE   C%WRITEA      IMPLANTATION    F58
                 MODULE   C%WRITEB      IMPLANTATION    F90
                 MODULE   OPCLTYP       IMPLANTATION    FE8
                                             LONGUEUR DU SEGMENT 10F8

     SEGMENT     CALCU%00  NO    2      IMPLANTATION   10F8
                 MODULE   CALCUL00      IMPLANTATION      0
                                             LONGUEUR DU SEGMENT  158
```

Programul 21.2 (continuarea a 2-a)

În geografia fişierelor sînt recapitulate datele de bază care caracterizează fiecare fişier, şi se indică şi adresa relativă a TDF.

Ariile tampon se găsesc în segmentele de comun corespunzătoare fiecărui fişier.

21.29 *Lista programului obiect* (*OBL*) se poate obţine specificînd opţiunea OBL în cartela COMPILE. Lista permite localizarea instrucţiunii care a produs întreruperea programului ca şi examinarea soluţiilor compilatorului în traducerea programului sursă în cod maşină, ceea ce poate duce la alegerea variantelor mai eficiente.

Pentru a afla instrucţiunea care a cauzat întreruperea se scade din adresa absolută (ETAT PROGRAMME) a instrucţiunii conţinutul registrului R14 şi adresa de lansare (IMPLANTATION) dată de editorul de legături pentru segmentul program respectiv.

ADR. ETAT PROGRAMME — R14 — IMPLANTATION — 4

Adresa astfel găsită se identifică în OBL coloana CPT.ORDI. Rîndul conţine o instrucţiune în cod maşină (INSTRUCTION) şi decodificarea în ASSIRIS (INSTRUCTION DECODEE). În stînga se specifică linia din programul sursă (LIG.SOURC) şi verbul (CODE VERBE) din care a provenit instrucţiunea.

În cazul în care adresa astfel găsită este exterioară programului COBOL propriu-zis, de cele mai multe ori este util să ne referim la datele din modulul de legături SGF.

21.30 *Î*. Ne propunem depanarea *exemplului de program 21.1* din cadrul acestei lecţii.
Programul s-a oprit cu mesajul . . . .
care înseamnă . . . .

R. ERREUR PROGRAMME 204, violarea unei partiţii protejate (v. anexa 2).

21.31 *Î*. Pentru a afla în ce segment s-a produs eroarea căutăm tabelul de segmente, care începe la adresa . . . . . aflată în registrul . . . .

R. 250BC, R15 (v. 21.23, 21.26).

21.32 *Î*. Adresa segmentului curent se află în registrul . . . . . . şi este egală cu . . . . . . Căutînd în tabelul de segmente aflăm că întreruperea s-a produs în segmentul nr. . . avînd numele . . . .

R. R14 (v. 21.23), 250C8, 1, TEST1%00(NO.1) (v. în mesajele editorului de legături pentru exemplul 21.1).

21.33 *Î*. Instrucţiunea care a cauzat întreruperea se află la al . . . .-lea octet (în hexazecimal) din segment, deci modulul . . . . deoarece acest modul este plasat între octeţii . . şi . .

R. 50(25118—250C8), TEST1%01, 20, 60.

21.34 *Î*. Instrucţiunea are, în ASSIRIS, mnemonicul ... şi a fost generată din enunţul COBOL (verbul) . . . corespunzător rîndului . . . din programul sursă.

R. 1. CYBR (recopiere de la stînga spre dreapta) (din 50—20 = 30)*, MOVE, 17 (v. 21.29).

21.35 *Î*. Bănuim că în cadrul acestui enunţ indicele este incorect. Ne propunem să examinăm conţinutul cîmpului corespunzător. Ne referim

---

* În mod obişnuit se mai scad 4 octeţi, dar în cazul unor erori de adresare folosim chiar valoarea dată în ETAT PROGRAMME.

la lista . . . . şi în cadrul ei la secţiunea . . . . Cîmpul I are adresa relativă . . . . faţă de adresa de început a segmentului de date, adresă care se găseşte în . . . . şi este egală cu . . . .

R. MAP (GEOGRAPHIE DES DONNEES), WORK (WORKING–STORAGE), 46 (v. 21.28), R13 sau în tabelul de segmente, 26558 (v. 21.23, 21.26).

21.36 *I.* Deci cîmpul I se află plasat în memorie, de la adresa . . . . . . pe o lungime de . . octeţi. Căutăm în vidaj acest cîmp. Constatăm că el conţine valoarea . . . . .

R. 2659E (26558 + 46) (v. 21.31), 9 (v. şi progr. sursă) 999999999.

21.37 *I.* Întrucît tabloul declarat în memorie nu are decît . . . elemente valoarea indicelui, găsită mai sus, era firesc să ducă la o adresă aberantă. Soluţia este . . . .

R. 100 (v. progr. sursă), o testare a valorii indicelui imediat după ACCEPT.

21.38 *I.* Propunem spre examinare, în continuare *exemplul de program 21.2* din cadrul acestei lecţii. S-a constatat o întrerupere a programului, avem la îndemînă doar lista de la imprimantă, pe aceasta nu apare nici un mesaj de eroare. Pentru a afla în ce segment s-a produs întreruperea, căutăm în tabelul de segmente. Adresa de întrerupere este . . . . şi ea corespunde segmentului nr. . . cu numele . . . .

R. 26474 (din ETAT PROGRAMME, v. 21.26), 1 (v. 21.29), TEST2%00 (mesajele ed. de legături).

21.39 *I.* Pentru a afla deplasarea (adresa relativă), deci modelul în care s-a produs întreruperea scădem adresa . . . . de la început a segmentului din adresa de întrerupere, obţinînd . . . . . . Constatăm că întreruperea s-a produs în afara modulului . . . generat din programul sursă COBOL, care este cuprins între adresele relative . . . şi . . . .

R. 250E0, 1394, TEST2%01, 38, 1C8, (vezi mesajele ed. de legături şi 21.29).

21.40 *I.* Bănuind că de vină sînt instrucţiunile referitoare la fişiere, ne propunem să examinăm rezumatul SGF. Adresa de început a acestuia se află în . . . . , în octeţii . . . , şi este egală cu . . . .

R. ZIG, 4—7, 250B0, (v. 21.27)

21.41 *Î.* Constatăm că programului i-au fost alocate . . fișiere în rezumatul SGF față de . . fișiere declarate în programul sursă

R. 3 (v. 21.28), 2

21.42 *Î.* Specificați idex-ul fișierelor prevăzute în rezumatul SGF (în hexazecimal și caractere), dacă sînt de intrare (INPUT) sau ieșire (OUTPUT) și dacă sînt sau nu deschise. Identificați fișierele în programul sursă.

R. 5C/—intrare deschis (SYSIN—FISCART)
5C/—ieșire nedeschis (SYSIN— ? ! ?)
C2/B—ieșire deschis (FISBANDA)
(v. 21.25 și programul sursă)

21.43 *Î.* Cele de mai sus ne conduc la examinarea verbelor READ/WRITE din programul sursă. Care este eroarea?

R. Enunțul WRITE CARTELA este eronat. Încercarea de a scrie CARTELA (asociată cu SYSIN) duce la declararea de către compilatorul COBOL a unui fișier aberant, al doilea dintre cele din rezumatul SGF.

21.44 Pentru realizarea unor programe corecte în condiții optime s-au elaborat o serie de *tehnici de testare și validare a programelor.* Un numitor comun al acestora este necesitatea unui *plan de testare* care va conține între altele :

— condițiile în care se consideră că un modul, un program sau un flux de programe este corect

— datele cu care se rulează (date de test și reale) cum se obțin acestea, ce condiții trebuie să îndeplinească

— fișiere de test, cine le elaborează, ce dimensiuni vor avea, la ce aplicații se mai folosesc

— succesiunea și calendarul pentru testarea modulelor, programelor și a întregului sistem, cu date de test și date reale

— responsabilități

Unele metodologii recomandă construirea întregului sistem de la bun început, din module „vide" și verificarea funcționării ansamblului. Pe măsură ce anumite module se realizează ele înlocuiesc modulele vide corespunzătoare, fiind testate imediat și condițiile integrării în sistem.

În cele ce urmează schițăm cîteva recomandări privind următoarele aspecte ale testării programelor :

— testarea schemelor logice și de structură
— codificarea preventivă a programelor
— testarea manuală a programelor
— testarea cu calculatorul
— modificarea programelor.

21.45 *Testarea schemelor logice și de structură.*

Trebuie făcută în mod obligatoriu înainte de a începe scrierea în limbaj de programare. Se verifică existența și corectitudinea pentru :

— inițializări, prelucrări, terminări
— structurile cu caracter repetiv, ciclările
— condițiile logice
— valorile variabilelor (comutatori, parametri, etc.) la intrările și ieșirile subprogramelor
— relațiile reciproce între fișiere și înregistrări, etc.

Pentru verificări se construiesc în prealabil mici fișiere și alte date de test, simulînd pe hîrtie mersul prelucrării. Se va realiza parcurgerea tuturor ramurilor schemei. Metoda rămîne valabilă și pentru tehnici alternative la schemele logice, ca de pildă în cazul utilizării pseudocodului.

21.46 Codificarea, adică scrierea programului în conformitate cu regulile limbajului de programare, trebuie să fie astfel făcută încît să reducă pe cît posibil probabilitatea apariției erorilor. Printre cele mai importante recomandări în acest sens amintim:

— evitarea denumirilor fanteziste pentru date, înregistrări, fișiere, paragrafe etc. De pildă, o anumită variabilă va avea același nume (nume cît mai sugestiv) în toate fișierele, adăugîndu-se ca prima literă idex-ul fișierului, iar în secțiunea WORKING—STORAGE un W.

— introducerea a cît mai multe comentarii (de preferință cu * în col. 7) atît în definirea cîmpurilor de date cît și în diviziunea PROCEDURE, ori de cîte ori credem că instrucțiunile codificate nu pot fi înțelese la prima citire de un alt programator. Se recomandă de asemenea folosirea numelor de condiție (nivel 88).

— evitarea artificiilor de programare și folosirea soluțiilor standard. Utilizarea unei structuri cît mai strict modulare (vezi lecția 20).

— utilizarea verbului PERFORM cu eliminarea pe cît posibil a lui GO TO. Un paragraf apelat cu PERFORM nu este permis să mai fie apelat și prin GO TO sau executat prin trecere directă. Excluderea cu desăvîrșire a verbului ALTER.

— realizarea ciclărilor prin PERFORM ... UNTIL ... sau, în cazul în care în ciclare este implicată variația unui indice, prin PERFORM ... VARYING ... UNTIL ..., avînd drept condiție: indice > nr. maxim. Ar trebui evitată varierea indicilor altfel decît prin acest mod, sau, eventual, prin SEARCH.

— evitarea condițiilor compuse care pot duce la ambiguități (de ex. cu forme abreviate și NOT). Evitarea în general a condițiilor compuse în cazul în care programatorul nu stăpînește elementele algebrei booleene (regulile cu AND, OR, NOT, cum sînt de pildă legile lui De Morgan).

— evitarea construcțiilor cu IF ... ELSE ... multiplu, care pot fi sursă de erori mai ales în perioada introducerii de modificări.

— evitarea utilizării unor cîmpuri (mai ales comutatori sau indici), în scopuri multiple. Adesea o economie în general nesemnificativă de memorie duce la confuzii mari și pierdere însemnată de timp calculator pentru depanare.

— evitarea utilizării unor posibilități COBOL insuficient înțelese, de ex. SEARCH sau SET în lucrul cu tablourile sau COMP în condiții care pun restricții de aliniere, cum ar fi înregistrările blocate.

— asigurarea unui aspect aerisit ; de exemplu să nu se mai scrie nimic într-un rînd în care s-a declarat un titlu de paragraf, un rînd să nu conţină mai mult de un verb, structurile ierarhizate să fie realizate printr-o retragere corespunzătoare faţă de col. 12.

— introducerea unor mesaje de eroare complete şi care se autodefinesc.

21.47 *Testarea manuală a programelor* se execută după codificare şi înainte de prima compilare. Se aseamănă cu testarea schemelor (v. 21.45). Programatorul trebuie să verifice între altele :

— iniţializările şi terminările
— deschiderea/închiderea tuturor fişierelor
— poziţia comutatoarelor
— testele (cu IF) şi completitudinea lor, ieşirile pentru ambele alternative
— contoarele şi indicii în ciclări
— logica succesiunii componentelor în execuţie
— corectitudinea cu care s-au transpus algoritmii de calcul, factorii de scară eventuali
— completitudinea mutărilor, mai ales la componentele înregistrărilor.

21.48 *Testarea cu calculatorul* se face într-o succesiune determinată (v. 21.44), ţintind la început eliminarea erorilor de sintaxă apoi a celor de logică şi de la simplu spre complex. La început se vor folosi date de test care să permită testarea tuturor ramurilor avînd în acelaşi timp volum redus şi permiţînd confruntarea rezultatelor date prin program cu rezultate calculate manual, pe baza aceloraşi valori. În cazul unor programe de validare, va trebui ca datele de test să producă toate mesajele de eroare şi cîteva combinaţii semnificative ale acestora. Se recomandă să se utilizeze :

— metodele automate de testare (DBG, PRINT, etc.)
— controlul numărului de înregistrări (tranzacţii, rînduri la imprimantă, etc.)
— totaluri de control
— controlul timpului de prelucrare (de ex. prin TIME în cartela RUN).

21.49 *Modificarea programelor* poate deveni o sursă serioasă de noi erori dacă nu este făcută în mod corespunzător.

Modificarea programelor este inevitabilă atît în timpul testării cît şi mai tîrziu. Este bine ca să existe o procedură pentru schimbare conţinînd o cerere tipizată de schimbare, aprobarea ei, modificarea programelor, (sursă, IMT etc.) implicate şi a documentaţiei aferente.

Erorile constatate trebuie imediat înlăturate, totuşi nu înainte de a cîntări toate consecinţele posibile ; se constată foarte adesea că modificările pripite introduc noi erori. Din acelaşi motiv nu se recomandă corectarea programelor în sala calculatorului.

LECȚIA a 22-a PROGRAME UTILITARE

- noțiuni și rol
- programul INVFICH
- programul MAINT

22.1 Programele utilitare sînt părți componente ale sistemului de operare. Pot fi apelate utilizînd cartela de comandă SYSRUN sau RUN din biblioteca Utilizator Standard (BUS) în scopul executării unor prelucrări frecvent întîlnite în rularea aplicațiilor pe calculator.

Pentru precizarea condițiilor de rulare se utilizează cartelele de comandă cu procent ('%') în coloana 1.

Pentru execuția unui program utilitar trebuiesc specificate informații referitoare la:

— numele programului utilitar;
— ce funcție se dorește să se execute;
— informații referitoare la volum, la etichete de fișier, la condițiile în care se dorește să se desfășoare execuția și opțiuni asupra execuției ce sînt caracteristice fiecărui program utilitar.

Sfîrșitul informațiilor referitoare la condițiile de execuție ale programului utilitar este anunțat de întîlnirea cartelei % END (cu excepția programului bibliotecar pentru care această cartelă este % ENDLIB).

22.2 Vom prezenta modul de utilizare pentru unele funcțiuni ale programelor utilitare:

— INVFICH (program de inventariere fișiere);
— MAINT (programe de întreținere a fișierelor).

Aceste programe utilitare sînt programe interpretative.

Mai există o categorie de programe utilitare cum ar fi SMGEN (program de sortare și interclasare) și CONVGEN (program de conversie de suporturi) care permite generarea de programe pentru sortare/interclasare, respectiv pentru conversii.

Apelarea acestor programe, cu cartela SYSRUN sau RUN urmată de cartelele ce precizează informații și opțiuni asupra execuției, permite asamblarea unui program în care se pot introduce și secvențe de program ale utilizatorului, efectuîndu-se automat executarea editării de legături. Programul generat astfel obținut poate fi catalogat într-o bibliotecă IMT sau încărcat pentru execuție prin intermediul unei cartele de comandă cu punct RUN.

22.3 *?*. Ce sînt programele utilitare, unde sînt memorate și cum se face apelarea lor?

R. Programele utilitare reprezintă facilităţi ale sistemului de operare în executarea simplă şi rapidă a unor prelucrări des întîlnite. Se găsesc în biblioteca Utilizator Standard (BUS) şi pot fi apelate prin cartelele SYSRUN sau RUN pe care este precizat numele programului respectiv.

22.4 *Î.* Ce deosebire este în ce priveşte execuţia între un program generat şi unul interpretativ?

R. Un program utilitar generat poate fi catalogat într-o bibliotecă utilizator IMT sau executat iar un program interpretativ este numai executat.

22.5 *Î.* Cum este precizat sfîrşitul cartelelor cu informaţii şi opţiuni referitoare la execuţia unui program utilitar?

R. Sfîrşitul cartelelor cu informaţii şi opţiuni referitoare la execuţia unui program utilitar este determinat de prezenţa cartelei % END iar pentru bibliotecar % ENDLIB.

22.6 *Programul utilitar INVFICH* îşi stabileşte condiţiile de execuţie utilizînd informaţiile şi parametrii indicaţi de utilizator şi permite obţinerea unor liste cu informaţii referitoare la rezumarea volumelor sau la inventarierea fişierelor. Programul poate trata volume de bandă magnetică organizate sau nu standard şi discuri magnetice prin intermediul funcţiunii RESUME şi fişiere organizate secvenţial, secvenţial-indexat, selective sau nedefinite cu funcţiunea DUMP.

22.7 Pentru execuţia funcţiei de rezumare a volumelor sînt necesare informaţii referitoare la funcţiunea de executat (apelată prin cartela RESUME) şi informaţii de descriere a volumului (conţinute în cartela DVOL).

Formatul cartelei RESUME:

% RESUME $\left\{ \begin{matrix} STD \\ \text{NST} \end{matrix} \right\}$, Mnumăr1, Bnumăr2

22.8 Argumentele au următoarea semnificaţie:

- *STD* — suport standard
- *NSD* — suport nestandard (bandă)
- *Mnumăr1* — *Bnumăr2* — reprezintă condiţiile de oprire
- *număr1* — numărul de mărci de fişier (file mark) citite, este un număr zecimal de la 0 la 99
- *număr2* — numărul de blocuri de citit după ultima marcă de fişier citită, este un număr zecimal de la 0 la 9999.

22.9 Formatul cartelei DVOL:

% DVOL $\left\{ \begin{matrix} \text{DV : periferic} \\ \text{DVT : tip—periferic, VS:ident— volum} \end{matrix} \right\}$

[,ON :'nume—proprietar']

$$[,VP : \text{'cuvînt'}] \left[,DVF : \left(\begin{Bmatrix} BIN \\ BCD \end{Bmatrix}, \begin{Bmatrix} TR9 \\ TR7 \end{Bmatrix} \right) \right]$$

cuvînt — cuvînt de acces la volum, format din 1 la 10 caractere alfanumerice

DVF — este un argument specific benzilor magnetice, precizînd felul înregistrării şi tipul de bandă (cu 9 sau 7 piste).

22.10 **Exemple:**

Inventarierea unui volum disc:

```
·   SYSRUN INVFICH
% RESUME
% DVOL DV:AD1
% END
```

Inventarierea unui volum bandă:

```
·   SYSRUN INVFICH
% RESUME
% DVOL DVT:MT,VS:MTMAN1
% END
```

22.11 *Î.* Se doreşte să se obţină rezumatul a două volume de bandă magnetică aflate pe MT1 (cu VS: MTMAN1) şi respectiv pe MT3 (cu VS: MTMAN 3). Cum se poate obţine prin intermediul programului INVFICH?

R.
```
SYSRUN INVFICH
% RESUME
% DVOL DV:MT1
% RESUME
% DVOL DVT:MT, VS:MTMAN 3
% END
```

Observaţie: s-a presupus că benzile magnetice au organizare standard (STD), au înregistrări binare (BIN) şi sînt cu 9 piste (TR9).

22.12 Apelarea funcţiei de inventariere a fişierelor este realizată cu ajutorul cartelei DUMP. Se produce afişarea la imprimantă a conţinutului de detaliu al unui fişier după formatul standard acceptat de program sau după un format precizat de utilizator. Informaţiile pot fi afişate în hexazecimal (cod EDCDIC) sau alfanumeric. Informaţiile sînt afişate împreună cu un antet ce conţine informaţii referitoare la bloc, sector, articol în scopul precizării originii informaţiilor.

Informaţiile de etichetă ale fişierului şi condiţiile de începere a afişării sînt descrise în cartela DFLE. Cu ajutorul cartelei KEY se precizează poziţia informaţiilor descrise prin cartelele DUMP şi DFLE. Opţiuni asupra formatului de afişare a informaţiilor se pot preciza prin cartelele LINE 1 şi LINE2.

Formatul cartelei DUMP

```
        ⎧      ⎡   ⎧ ⎧SEQ⎫ ⎡  ⎧EOF                                  ⎫⎤⎤⎤     ⎫
        ⎪      ⎢   ⎪ ⎩INS⎭ ⎢ ,⎨(⎧A cheie-alfanumerică ⎫...)          ⎬⎥⎥⎥     ⎪
        ⎪      ⎢   ⎪       ⎢  ⎪ ⎩X cheie-hexazecimală ⎭              ⎪⎥⎥⎥     ⎪
        ⎪      ⎢   ⎪       ⎣  ⎩Rnumăr1                              ⎭⎦⎥⎥     ⎪
        ⎪ STD  ⎢ , ⎨                                                  ⎥⎥ ⎡  ⎧AX⎫⎤⎪
% DUMP  ⎨      ⎢   ⎪ UND ⎡ , ⎧EOF      ⎫⎤                            ⎥⎥ ⎢ ,⎨A ⎬⎥⎬
        ⎪      ⎢   ⎪     ⎣   ⎩BLnumăr2 ⎭⎦                            ⎥⎥ ⎣  ⎩X ⎭⎦⎪
        ⎪      ⎢   ⎪                                                  ⎥⎥         ⎪
        ⎪      ⎢   ⎪ RAN ⎡ , ⎧EOF      ⎫⎤                            ⎥⎥         ⎪
        ⎪      ⎣   ⎩     ⎣   ⎩Cnumăr3  ⎭⎦                            ⎥⎥         ⎪
        ⎩NSTMnumăr4,Bnumăr5                                           ⎦⎦         ⎭
```

(STD, SEQ, EOF and AX are underlined as the default values.)

22.13 Argumentele au următoarele semnificații:

- *STD* — suport standard
- *SEQ* — fișier organizat secvențial
- *INS* — fișier organizat secvențial-indexat
- *UND* — fișier cu organizare nedefinită
- *RUN* — fișier cu organizare selectivă
- *EOF* — afișarea se execută pînă la sfîrșitul fișierului
- *A—cheie—alfanumerică, X—cheie—hexazecimală* = indică condiții de terminare a afișării conținutului de informații, pot indica o succesiune de caractere pînă la maxim 255 octeți (caracterele hexazecimale se reprezintă prin 2 caractere pe octet)
- *număr*1 — număr de articole de afișat, este un număr zecimal de la 1 la 999999
- *număr*2 — afișarea se execută numai pînă la un anumit bloc precizat prin număr2 care este un număr zecimal de la 1 la 99999
- *număr3* — afișarea se realizează pe un anumit număr de casete precizat prin numărul zecimal număr3;
- *AX, X, A* — forma de afișare care poate fi:

  AX — linia în hexazecimal este urmată de o linie cu aceeași informație în alfanumeric

  A — forma alfanumerică (1 caracter/octet)

  X — forma hexazecimală (2 caractere/octet)

22.14 *Formatul cartelei DFLE*:

— pentru suport standard

```
% DFLE ⎧ DV :periferic—n                          ⎫
       ⎩ DVT :tip—periferic,VS : ident—volum]     ⎭
,FN : 'nume—fișier [/nume—zonă]' [,UN : număr—actualizare]
```

$$\left[,\left\{\begin{matrix}\text{GN : număr—generaţie}\\ \text{CD dată—creare}\end{matrix}\right\}\right][,\text{VN număr—versiune}]$$

[,ON : 'nume—proprietar'] [,VP : 'cuvînt'] [,FP : 'cuvînt—fişier']

$$[,\text{DVF}:\left(\left[\left\{\begin{matrix}\underline{\mathit{BIN}}\\ \text{BCD}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\mathit{TR9}}\\ \text{TR7}\end{matrix}\right\}\right]\right)\Bigg]\Bigg[,\ \text{MFF}:\left(\left\{\begin{matrix}\underline{\mathit{NCB}}\\ \text{CBS}\end{matrix}\right\}\right)\Bigg]$$

$$\left[,\text{BEG}:\left(\left\{\begin{matrix}\text{Z număr1, Bnumăr2}\\ \left\{\begin{matrix}\text{A cheie—alfanumerică}\\ \text{X cheie—hexazecimală}\end{matrix}\right\}..\\ \text{Cnumăr3}\end{matrix}\right\}\right)\right]$$

$$\left[\text{OFO}:\left(\left[\left\{\begin{matrix}\underline{\mathit{RLS}}\\ \text{NRL}\end{matrix}\right\}\right]\right)\right]\left[,\text{CFO}:\left(\left[\left\{\begin{matrix}\underline{\mathit{RLS}}\\ \text{NRL}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\mathit{RWD}}\\ \text{DMT}\\ \text{NRW}\end{matrix}\right\}\right]\right)\right]$$

$$\left[,\text{CVO}:\left(\left\{\begin{matrix}\underline{\mathit{RWD}}\\ \text{DMT}\\ \text{NRW}\end{matrix}\right\}\right)\right]$$

pentru suport nestandard (numai bandă)

$$\%\ \text{DFLE DV:MTn}\left[,\text{DVF}:\left(\left[\left\{\begin{matrix}\underline{\mathit{BIN}}\\ \text{BCD}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\mathit{TR9}}\\ \text{TR7}\end{matrix}\right\}\right]\right)\right],\text{BFS : număr4}$$

$$[,\text{BEG : (Mnumăr5, Bnumăr6)}]\left[,\text{CFO}:\left(\left[\left\{\begin{matrix}\underline{\mathit{RLS}}\\ \text{NRL}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\underline{\mathit{RWD}}\\ \text{DMT}\\ \text{NRW}\end{matrix}\right\}\right]\right)\right]$$

22.15 Argumentele au următoarea semnificaţie

- *MFF* : $\left(\left\{\begin{matrix}\underline{\text{NCB}}\\ \text{CBS}\end{matrix}\right\}\right)$ se referă numai la fişierele secvenţiale pe bandă,

  NCB — fără caractere de control de secvenţă a blocurilor
  CBS — controlul secvenţei blocurilor

- *BEG* — precizează nivelul fizic de la care programul trebuie să înceapă prelucrarea :

  număr1 — numărul primei zone de prelucrat, între 1 şi 99
  număr2 — numărul primului bloc de prelucrat, număr zecimal cu valoare de la 1 la 99999

- *OFO* — precizează condiţiile de deschidere de fişier. numai pentru benzile de organizare standard :

  RWD — rebobinare înainte de deschidere
  NRW — păstrarea poziţionării

- *CFO* — precizează condiţiile de închidere de fişier
  RLS — eliberarea perifericului
  NRL — păstrarea afectării perifericului
  RWD — rebobinarea ultimei bobine prelucrate
  DMT — rebobinarea şi demontarea ultimei bobine prelucrate
- *CVO* — condiţii de închidere volum:
  RWD — rebobinarea bobinelor intermediare
  DMT — rebobinarea şi demontarea bobinelor intermediare
  NRW — păstrarea poziţionării bobinelor intermediare
- *număr5* — numărul de mărci de fişier (file mark-uri) după care începe prelucrarea
- *număr6* — numărul blocului de unde începe prelucrarea după ce s-au citit număr5 mărci de fişier

22.16 *Formatul cartelei KEY:*

% KEY adresă1, lungime1 [,adresă2,lungime2] ...

- adresă — precizează adresa relativă a cheii în raport cu începutul articolului, număr zecimal cu valoarea maximă 32767 (primul octet al articolului are adresa zero)
- lungime — indică dimensiunea cheii în octeţi

Sînt admise maxim 12 chei dimensiunea totală poate avea maxim 256 octeţi.

22.17 *Formatul cartelei LINE:*

$$\%\ \text{LINE}\begin{Bmatrix}1\\2\end{Bmatrix}\text{Cnumăr1,S}\begin{Bmatrix}0\\1\\2\\3\end{Bmatrix}\text{[,Rnumăr2] ,Z(Knr—octeţi [,Icaract—}$$

inserate] [,Rnumăr—zonă]) [,Z(Knr—octeţi [,Icaract—inserate] [,Rnumăr—zonă])] ...

unde:

- *număr1* — reprezintă numărul coloanei de început de linie, este un număr zecimal între 1 şi 99, iar dacă este mai mic decît lungimea antetului +1, antetul figurează singur într-o linie
- *S* — specifică saltul ce se efectuează înaintea imprimării liniei
- *R* — *număr2* — defineşte numărul de repetări pentru întregul format descris de cartelă, dacă se utilizează şi LINE2 atunci număr2 este egal cu 1; număr2 este un număr zecimal de 2 cifre
- *Z* — defineşte o zonă a formatului de afişare prin:
  - nr-octeţi — număr de caractere alfanumerice de imprimat de la 0 la 99, dacă afişarea se face în hexazecimal acest număr este multiplicat cu 2
  - *caract—inserate* — caractere alfanumerice de inserat cu rol de separatori după zonă
  - *număr—zonă* — număr de repetare al zonei din linie, are valori între 1 şi 10.

Dacă nu se precizează LINE $\left\{\frac{1}{2}\right\}$ implicit se consideră următoarele valori pentru :

• *LINE1* — lungimea antetului este funcție de tipul fișierului (număr 1 = 20), saltul se execută la un rînd (S1) K are valoarea nr—octeți = 10, se inserează un caracter spațiu iar R are o valoare pentru număr — zonă cuprinsă între 1 și 5

• *LINE2* — C are număr1 egal cu 1 iar în rest este la fel ca LINE1.

22.18 Sfîrșitul parametrilor este anunțat de cartela END care are formatul :

% END

22.19 **Exemple :**

1. Se dorește să se obțină conținutul fișierului FTRANZ aflat pe o bandă magnetică cu formatul de editare mixt (AX—valoare implicită)

```
•     SYSRUN INVFICH
% DUMP
% DFLE DV : MT3,FN : 'FTRANZ'
% END
```

2. Se dorește obținerea rezumatului unui disc (AD1) și să se afișeze înregistrările unui fișier secvențial-indexat cu FN : 'MATERIALE' începînd cu înregistrarea cu cheia de valoare 713555, cu formatul de editare alfanumeric.

```
•     SYSRUN INVFICH
% RESUME
% DVOL DV : AD1
% DUMP INS,A
% DFLE DV : AD1,FN : 'MATERIALE',BEG : (A713555)
% END
```

22.20 *Î.* Un fișier pe bandă cu FN : 'ARTICOLE' are înregistrări cu lungimea 90 octeți. Se dorește să se listeze 100 de înregistrări în format alfanumeric. Informațiile să fie grupate cîte 15, caracterele de inserat =/=. Care sînt cartelele ce trebuiesc specificate și ce informații conțin?

R.

```
·     SYSRUN INVFICH
% DUMP A,R100
% DFLE DV : MT3,FN : 'ARTICOLE'
% LINE1, C20,S3,Z(K15,I=/=,R6)
% END
```

22.21 Programul utilitar MAINT aeste un program interpretativ care oferă utilizatorilor sistemului de operare anumite facilități în controlul și întreținerea datelor ce se găsesc pe suporturi magnetice sub formă de fișiere.

Vom prezenta facilitățile programului MAINT referitoare la : salvarea, restaurarea și reorganizarea fișierelor considerînd că acestea sînt utilizate frecvent. Programul mai oferă facilități referitoare la duplicarea (copierea) fișierelor/volumelor (cartela COPY), la compararea conținului fișierelor, volumelor (cartela COMPARE), la corectarea fișierelor la nivel de articol sau octet (cartela CORRECT). Utilizatorul precizează funcția dorită și opțiuni asupra execuției acesteia prin intermediul a trei tipuri de parametri :

— parametri care definesc suporturile, fișierele și volumele
— parametri referitori la funcția de întreținere

— parametri secundari care detaliază aspectele de realizare a funcției de întreținere specificate.

Cartelele de descriere a fișierelor și volumelor au același format pentru toate funcțiile programului MAINT.

22.22 *Cartela OLDVOL* definește volumul emitor sau primul volum de comparare și are formatul :

$$\%\ \text{OLDVOL}\begin{Bmatrix}\text{DV : periferic}-\text{n}\\ \text{DVT : tip}-\text{periferic,VS : ident}-\text{volum}\end{Bmatrix}$$

$$[\text{,ON : 'nume}-\text{proprietar'}]\quad[\text{,VP : cuvînt}]$$

$$\left[\text{,DVF}\left(\left[\left\{\frac{\mathit{BIN}}{\text{BCD}}\right\}\right]\left[,\left\{\frac{\mathit{TR9}}{\text{TR7}}\right\}\right]\right)\right]\left[\text{,BEG : Bnumăr}-\text{bloc}\right]$$

$$[\text{,BFS : zona}-\text{tampon}]$$

unde :

• *BEG* — definește originea prelucrării, număr—bloc reprezintă numărul primului bloc considerat și este un număr zecimal între 1 și 99999

• *BFS* — specifică dimensiunea zonei tampon, este un număr zecimal între 1 și 655535 și este obligatoriu de specificat la volumele cu organizare nestandard

22.23 *Cartela OLDFLE* descrie fișierul emitor sau primul fișier de comparare și are formatu :

$$\%\ \text{OLDFLE}\begin{Bmatrix}\text{DV : periferic}-\text{n}\\ \text{DVT : tip-perifric,VS : ident}-\text{volum}\end{Bmatrix}$$

$$\text{FN : 'nume}-\text{fișier[/nume}-\text{zonă]'}\left[,\begin{Bmatrix}\text{GN : număr}-\text{generare}\\ \text{CD : dată}-\text{creare}\end{Bmatrix}\right]$$

$$[\text{,VN : număr}-\text{versiune}]\ [\text{,UN : număr}-\text{actualizare}]$$

$$[\text{,ON : 'nume}-\text{proprietar'}]\ [\text{,VP : 'cuvînt'}]\ [\text{,FP : 'cuv nt}-\text{fișier'}]$$

$$[\text{,FS : ident}-\text{general}-\text{volum}]$$

$$\left[\text{,DVF :}\left(\left[\left\{\frac{\mathit{BIN}}{\text{BCD}}\right\}\right]\left[,\left\{\frac{\text{TR9}}{\text{TR7}}\right\}\right]\right)\right]\left[\text{,MFF :}\left(\left[\left\{\frac{\mathit{NCB}}{\text{CRS}}\right\}\right]\left[,\left\{\frac{\mathit{NBL}}{\text{BLC}}\right\}\right]\right)\right]$$

$$\left[\text{,BEG :}\left(\left\{\begin{matrix}\text{Znumăr1, Bnumăr2}\\ \begin{Bmatrix}\text{Acheie}-\text{alfanumerică}\\ \text{Xcheie}-\text{hexazecimală}\end{Bmatrix}\ldots\end{matrix}\right\}\right)\right]$$

[OFO : ...] [,CFO : ...] [,CVO : ...]

Parametrii cartelei OLDFLE au fost descriși în 22. 14 în formatul cartelei DFLE. Apare în plus în argumentul MFF :

NBL — nu se efectuează control al lungimii blocurilor

BLC — se efectuează controlul lungimii blocurilor

Cînd se utilizează argumentul BEG este obligatorie prezența unei cartele KEY.

22.24 *Cartela NEWVOL* permite definirea volumului receptor și are formatul:

$$\text{\%NEWVOL}\begin{Bmatrix}\text{DV periferic}-\text{n}\\ \text{DVT: tip}-\text{periferic,VS : ident}-\text{volum}\end{Bmatrix}$$

$$\text{[,ON'nume :}-\text{proprietar'][,VP :'cuvînt']}\left[\text{,DVF :}\left(\left[\begin{Bmatrix}\underline{\mathit{'BIN}}\\ \text{BCD}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{TR9}}\\ \text{TR7}\end{Bmatrix}\right]\right)\right]$$

$$\text{[,BEG : Bnumăr}-\text{bolc] [,VR] [,BFS zona}-\text{tampon]}$$

Parametrii au aceiași descriere ca la cartela OLDVOL descrisă în 22.22 iar argumentul VRF are semnificația:

VRF — controlul scrierii pe discuri.

22.25 *Cartela NEWFLE* definește fișierul receptor sau al doilea fișier de comparat și are formatul:

$$\text{\% NEWFLE}\begin{Bmatrix}\text{DV : periferic}-\text{n}\\ \text{DVT : tip}-\text{periferic,VS : ident}-\text{volum]}\end{Bmatrix}$$

$$\text{[,FN : 'nume}-\text{fișier [/nume}-\text{zonă]' [,GN : număr}-\text{generare]}$$

$$\text{[,VN : număr}-\text{versiune] [,UN număr}-\text{actualizare]}$$

$$\text{[,FS : ident}-\text{general}-\text{volum] [,ON : 'nume}-\text{proprietar']}$$

$$\text{[,VP : 'cuvînt'] [,FP : 'cuvînt}-\text{fișier']}\left[\text{,DVF :}\left(\left[\begin{Bmatrix}\underline{\mathit{BIN}}\\ \text{BCD}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{TR9}}\\ \text{TR7}\end{Bmatrix}\right]\right)\right]$$

$$\left[\text{,MFF :}\left(\left[\begin{Bmatrix}\underline{\mathit{NBL}}\\ \text{BLC}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{NCB}}\\ \text{CBS}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{NDC}}\\ \text{DLC}\end{Bmatrix}\right]\right)\right]$$

$$\left[\text{,AM :}\begin{Bmatrix}\text{ANY}\\ \text{NEW}\\ \text{OFL}\end{Bmatrix}\right]\text{[,SZ :nr}-\text{cilindri]}\left[\begin{Bmatrix}\text{OG : nr. generare1}\\ \text{OD : data}-\text{creare1}\end{Bmatrix}\right]$$

$$\text{[,OV: nr}-\text{versiune1]}\left[\text{,BEG :}\left(\begin{Bmatrix}\text{Z număr1 [,Bnumăr2]}\\ \begin{Bmatrix}\text{A cheie}-\text{alfanumerică}\\ \text{X cheie}-\text{hexazecimală}\end{Bmatrix}\end{Bmatrix}\right)\right]$$

$$\text{[BFS : zonă tampon]}$$

$$\left[\text{,OFO :}\left(\left[\begin{Bmatrix}\underline{\mathit{RWD}}\\ \text{NRD}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{BEG}}\\ \text{END}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{NVR}}\\ \text{VRF}\end{Bmatrix}\right]\right)\right]$$

$$\left[\text{,CFO :}\left(\left[\begin{Bmatrix}\underline{\mathit{RLS}}\\ \text{NRL}\end{Bmatrix}\right]\left[,\begin{Bmatrix}\underline{\mathit{RWD}}\\ \text{DMT}\\ \text{NRW}\end{Bmatrix}\right],\begin{Bmatrix}\underline{\mathit{NAD}}\\ \text{ADP}\end{Bmatrix}\right)\right]\left[\text{,CVO :}\left(\left[\begin{Bmatrix}\underline{\mathit{RWD}}\\ \text{DMT}\\ \text{NRW}\end{Bmatrix}\right]\right)\right]$$

majoritatea parametrilor se găsesc și în cartela OLDFLE și au fost descriși în formatul cartelei DFLE.

22.26 Argumentele au următoarea semnificaţie:

Parametrii OG, OD şi OV sînt utilizaţi dacă se specifică AM : OFS şi se referă la fişierul de eliberat.

- BEG — originea prelucrării
- END — descriere pentru scriere în continuare
- NVR — fără control la scriere
- VRF — control la scriere prin recitire şi comparare

22.27 *Funcţia de protecţie a informaţiilor* are două componente:

a) Salvarea informaţiilor (cartela SAVE):
a.1. salvarea unui fişier
a.2. salvarea unui volum (partea utilă)
b) Restaurarea informaţiilor (cartela REST):
b.1. restaurarea unui fişier
b.2. restaurarea unui volum

Atît salvarea cît şi restaurarea informaţiilor se poate efectua cu verificare (cartelele SCHECK respectiv RCHECK).

22.28 *Salvarea informaţiilor*—este realizată această funcţie la întîlnirea cartelei SAVE care efectuează o duplicare a unui fişier sau volum pe un suport diferit sau de acelaşi tip. Informaţiile obţinute prin duplicare nu pot fi exploatate decît după restaurarea lor pe un suport identic cu cel iniţial cu ajutorul cartelei REST.

22.29 *Formatul cartelei SAVE* este:

$$\%\ \text{SAVE}\left[\left\{\begin{matrix}\mathit{FLE}\\ \text{VOL}\end{matrix}\right.\right][\text{,BF : zonă—tampon}]$$

unde:

- *FLE* — presupune salvare fişier
- *VOL* — salvare volum (se salvează cilindrii ocupaţi)
- *BF* — prin zona—tampon se poate indica dimensiunea unităţii de informaţie de transferat (prin lipsă se consideră 5120 octeţi)

Formatul cartelelor SCHEK şi RCHECK este:

$$\%\ \text{SCHECK}\left[\left\{\begin{matrix}\mathit{FLE}\\ \text{VOL}\end{matrix}\right\}\right]\ \text{verificare la salvare}$$

$$\%\ \text{RCHECK}\left[\left\{\begin{matrix}\mathit{FLE}\\ \text{VOL}\end{matrix}\right\}\right]\text{verificare la restaurare}$$

22.30 *Format cartelă REST:*

Cu ajutorul cartelei REST se poate restaura în scopul exploatării (prelucrării) un fişier sau volum salvat prin SAVE.

Restaurarea se efectuează pe un suport identic cu suportul iniţial de pe care a fost salvat fişierul sau volumul.

$$\%\ \text{REST}\left[\left\{\begin{matrix}\mathit{FLE}\\ \text{VOL}\end{matrix}\right\}\right][\text{,BF : zonă—tampon}]$$

22.31 **Exemple:**

1. Să se scrie cartelele necesare apelării funcţiilor de salvare şi restaurare a unui volum de disc (AD1) pe o bandă, cu efectuarea verificărilor

a) Salvare volum

```
.    SYSRUN MAINT
% SAVE VOL
% OLDVOL DV:AD1
% NEWFLE DV:MT1,FN:'SALVDISC'
% SCHECK VOL
% END
```

b) Restaurare volum

```
.    SYSRUN MAINT
% REST VOL
% OLDFLE DV:MT1,FN:'SALVDISC'
% NEWVOL DV:AD1
% RCHECK VOL
```

2. Să se scrie cartelele necesare pentru salvarea cu verificare a unui fişier, respectiv restaurarea cu verificare a aceluiaşi fişier pe disc.

a) Salvare fişier

```
.    SYSRUN MAINT
% SAVE FILE
% OLDFLE DV:AD1,FN:'FISPERS'
% NEWFLE DV:MT3,FN:'PERSBAND'
% SCHECK
% END
```

b) Restaurare fişier

```
. SYSRUN MAINT
% REST
% OLDFLE DV:MT3,FN:'PERSBAND'
% NEWFLE DV:AD1,FN:'FISPERS',AM:ANY,SZ:15
% RCHECK
% END
```

22.32 *Î.* Se doreşte ca considerînd exemplele 22.31 să se scrie cartele numai pentru verificarea operaţiilor de salvare respectiv restaurare pentru un volum.

R.

```
.        SYSRUN MAINT
% SCHECK VOL
% OLDVOL DV:AD1
% NEWFLE DV:MT1,FN:'SALVDISC'
% END
.        SYSRUN MAINT
% RCHECK VOL
% OLDVOL DV;MT1,FN:'SALVDISC'
% NEWVOL DV:AD1
% END
```

22.33 *Funcţia de reorganizare a unui fişier* se referă la reorganizarea suportului ocupat de un fişier secvenţial-indexat (ISORG) sau de un fişier selectiv (RDORG).

22.34 Reorganizarea unui fişier secvenţial-indexat cu programul utilitar MAINT se efectuează prin citirea secvenţială a fişierului la nivel articol, după care se rescrie secvenţial fişierul citit.

Prin această operaţie sînt suprimate articolele şterse (cu instrucţiunea DELETE în COBOL) iar articolele din zona de depăşire sînt introduse în secvenţă.

Format :

% ISORG

Această cartelă nu conţine parametri.

Exemplu :

```
. SYSRUN MAINT
. ISORG
% OLDFLE DV : RD1,FN : 'FPERS1'
% NEWFLE DV : RD1,FN : 'FPERS2',AM : ANY,SZ : 55
% END
```

22.35 În cazul reorganizării unui fişier selectiv suporturile emiţător şi receptor folosite pentru reorganizare trebuie să fie discuri a căror dimensiune de sectoare şi număr de cilindri trebuie să fie identice.

Reorganizarea se efectuează la nivelul casetei.

Sînt citite toate articolele dintr-o casetă şi se suprimă articolele de şters.

Format :

% RDORG [DLC] [,LIST]

DLC — opţiune pentru ştergerea numărului de actualizare a articolelor

LIST — opţiune pentru afişarea articolelor fişierului

```
.      SYSRUN MAINT
% RDORG DLC,LIST
% OLDFLE DV : RD1,FN : 'CONTRACTE'
% NEWFLE DV : RD2,FN : 'RDCONTR',AM : ANY,SZ: 15
% END
```

22.36 *Î.* Un fişier secvenţial-indexat pe disc este salvat cu programul MAINT prin SAVE pe o bandă magnetică. Se poate prelucra fişierul salvat în condiţiile unui fişier secvenţial pe bandă ştiind că fişierul secvenţial indexat este citit secvenţial şi trecut pe bandă?

R. Nu se poate citi secvenţial pe bandă. Prelucrarea sa se poate efectua numai după ce a fost restaurat pe disc.

TESTUL 1 PRELUCRAREA AUTOMATĂ A DATELOR

Î. Un calculator se compune dintr-un ansamblu de echipamente cu funcții bine precizate cum ar fi: . . . .
. . . .
. . . .

R. Pentru verificarea răspunsului consultați Lecția 1.

Î. Datele și programele trebuie să se găsească în momentul prelucrării, în . . . .

R. memoria centrală a calculatorului.

Î. Scrierea programelor se realizează folosind un anumit . . . . .
. . . . pentru care există . . . .
inclus în programele sistemului de operare.

R. — limbaj de programare;
— un program traducător (compilator).

Î. Programul este o suită de . . . . date calculatorului și respectă structura (algoritmul) procesului de prelucrare Fiecare instrucțiune cuprinde două părți: prima care indică . . .
. . . . și a doua care indică . . . .
. . . .

R. — instrucțiuni (comenzi);
— tipul comenzii (instrucțiunii);
— operanzii (elementele la care se referă comanda).

Î. Un program poate cuprinde două mari categorii de instrucțiuni și anume . . . .
. . . .

R. — instrucțiuni de prelucrare (intrare/ieșire, calcul, transfer)
— instrucțiuni de direcționare a procesului de prelucrare către o instrucțiune sau alta, către o secvență sau alta (salt, test, apel) numite și instrucțiuni de control.

Î. În evoluția programării calculatoarelor electronice definim mai multe categorii de limbaje:
. . . .

. . . .
. . . .

R. pentru corectarea răspunsului consultaţi Lecţia 1.

Î. Limbajul COBOL face parte din categoria . . . . şi se particularizează prin facilităţile oferite pe linia organizării şi prelucrării datelor organizate în . . . .

R. — limbajelor evoluate;
— fişiere.

Î. Deoarece calculatorul cunoaşte doar . . . . . . este necesară existenţa unor programe traducătoare numite şi . . . . . . care realizează trecerea automată (dacă programele sînt corecte) de la forma . . . . . . (în limbaj evoluat), la forma . . . . (în limbaj maşină)

R. — limbajul maşină;
— compilatoare;
— program sursă;
— program obiect.

Î. Compilarea înseamnă: mai întîi efectuarea analizei . . . . pentru a delimita componentele programului; apoi a analizei . . . . . pentru a verifica dacă programul este sintactic corect şi în final . . . .

R. lexicale; de sintaxă; generarea programului obiect.

Î. Sistemul de operare este constituit dintr-un . . . . care au rolul de a . . . . Legătura între program şi sistemul de operare se asigură prin intermediul . . . .

R. — ansamblu de programe de bază (furnizat odată cu sistemul de calcul ;
— coordona întreaga activitate în procesul prelucrării automate a datelor;
— unor comenzi speciale (cartelelor de comandă).

Î. Realizarea unei lucrări (program) presupune parcurgerea unor etape bine precizate şi anume: . . . .
. . . .
. . . .
. . . .

R. — studierea problemei pe baza dosarului de analiză;
— proiectarea programului;

— codificarea sau scrierea pe formularul de programare COBOL;
— compilarea, testarea şi depanarea programului.

Î. Cînd datele sînt în volum mare, programatorul le organizează în . . . . folosind una din metodele de organizare . . . .
Organizările secvenţială-indexată şi selectivă nu sînt admise decît în condiţiile folosirii suporturilor. . . . .

R. — fişiere;
— secvenţială;
secvenţială-indexată;
selectivă;
— adresabile (discul magnetic).

Î. În principiu, organizările secvenţială-indexată şi selectivă urmăresc realizarea unei . . . . între valorile unei variabile discriminatorii (cheie) şi locaţia de memorie la care este . . . .

R. — corespondenţe
— memorat articolul în cauză (în general grupa de articole).

Î. Pentru a asigura securitatea şi controlul privind accesul la datele fişierelor pe suporturi magnetice, acestea sînt însoţite de articole speciale, numite . . . .

R. etichete
Răspunsurile dumneavoastră sînt corecte?
DA → Lecţia 3
NU ↺ Lecţia 1

TESTUL 2 PROGRAM DE LUCRU CU FIȘIERE PE CARTELE ȘI LA IMPRIMANTĂ

Să se elaboreze programul necesar obținerii unei situații la imprimantă utilizînd în intrare un fișier pe cartele cunoscînd:

— fluxul general al datelor (fig. T.2.1).

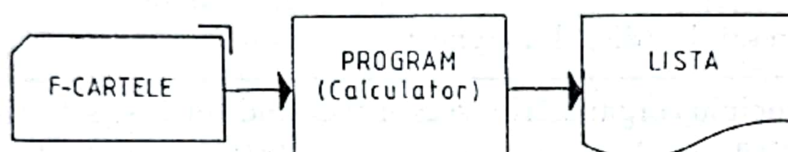


Fig. T.2.1.

— structura articolelor asociate celor două fișiere este redată în figura T.2.2.

GESTIUNE	SIMBOL MATERIAL	DENUMIRE MATERIAL	UNITATE DE MASURA	PRET UNITAR	STOC EFECTIV	STOC NORMAT
n	n	a-n	a-n	n 5+2	n 5+2	n 5+2
1	2 - 8	9 - 38	39 - 41	42 - 48	49 - 56	57 - 63

7 3 3

NRC	GEST.	SIMBOL MATERIAL	DENUMIRE MATERIAL	UNIT MAS.	PRET UNITAR	STOC EFECTIV	STOC NORMAT

TOTAL VALOARE STOC EFECTIV *****
TOTAL VALOARE STOCURI SUPRANORMATIVE *****

Fig. T.2.2.

Specificaţii de prelucrare:

— nu vor fi supuse prelucrării decît acele cartele care au fost corect perforate, pentru aceasta se va testa modul în care sînt îndeplinite sau nu condiţiile (cazul a fost mult simplificat):

— GESTIUNE ∈ (01—14);

— PREŢ, natură numerică;

— pentru toate articolele corecte se va înregistra cîte un rînd în situaţia stocurilor de materiale.

Întrucît la sfîrşit se cere furnizarea valorii materialelor aflate în stoc şi a valorii stocurilor supranormative se fac următoarele calcule;

— valoarea materialelor aflate în stoc;

VAL—ST—EF: = CANT—STOC—EF * PRET—UNITAR

— cumularea valorii pentru aflarea la sfîrşit a totalului;

T—VAL—ST—EF: = T—VAL—ST—EF + VAL—ST—EF

— valoarea stocurilor supranormative;

dacă

CANT—STOC—EF > CANT—STOC—N

atunci

VAL—SUT—SPR: = (CANT—STOC—EF — CANT—STOC—N)* PRET—UNITAR

— cumularea valorii pentru a afla la sfîrşit totalul

T—VAL—ST—SUPR; = T—VAL—ST—SUPR + VAL—ST—SUPR

Imaginea memoriei este cea din figura T.2.3.

Verificaţi răspunsul cu cel prezentat în fig. T.2.4. şi programul T.2.1. El este asemănător?

DA ↰ Lecţia 9

NU ↰ 8.1.

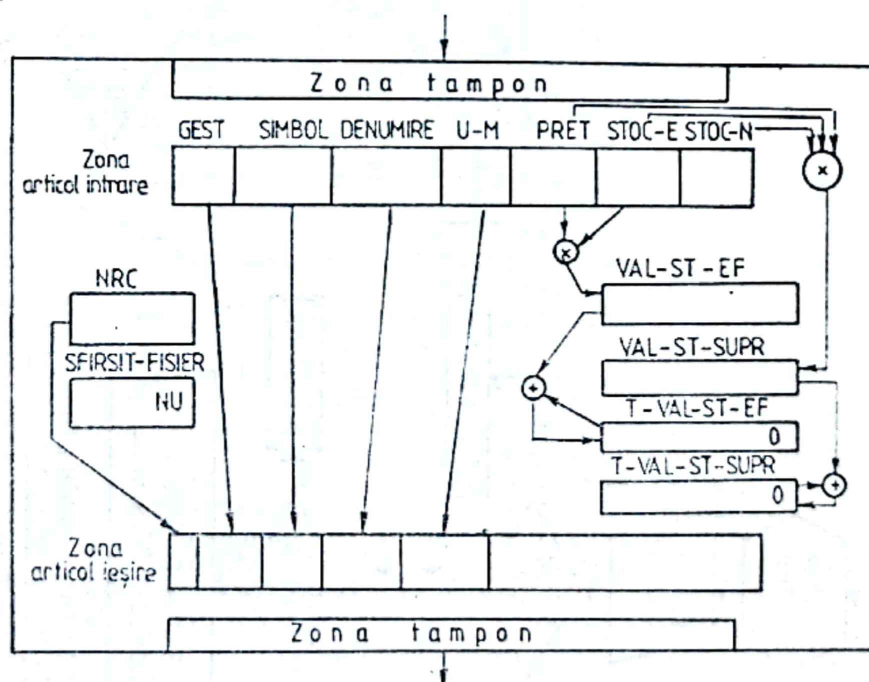


Fig. T.2.3.

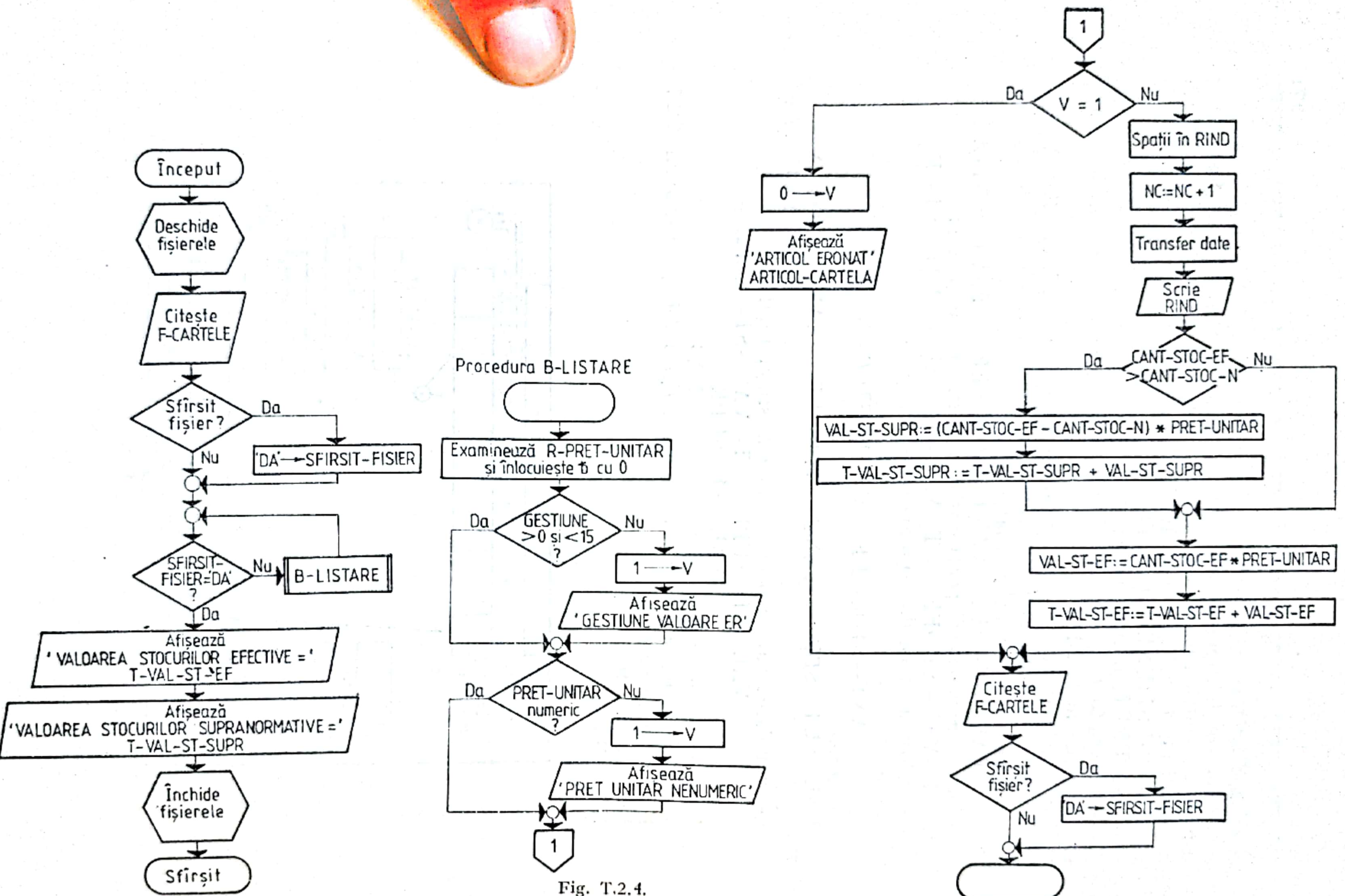

Început
Deschide fișierele
Citește F-CARTELE
Sfîrșit fișier?
Da
Nu
'DA' → SFIRSIT-FISIER
SFIRSIT-FISIER='DA' ?
Nu
B-LISTARE
Da
Afișează ' VALOAREA STOCURILOR EFECTIVE =' T-VAL-ST-EF
Afișează 'VALOAREA STOCURILOR SUPRANORMATIVE =' T-VAL-ST-SUPR
Închide fișierele
Sfîrșit
Procedura B-LISTARE
Examinează R-PRET-UNITAR și înlocuiește ƀ cu 0
GESTIUNE >0 și <15 ?
Da
Nu
1 → V
Afișează 'GESTIUNE VALOARE ER'
PRET-UNITAR numeric ?
Da
Nu
1 → V
Afișează 'PRET UNITAR NENUMERIC'
1
1
V = 1
Da
Nu
0 → V
Afișează 'ARTICOL ERONAT' ARTICOL-CARTELA
Spații în RIND
NC:=NC+1
Transfer date
Scrie RIND
CANT-STOC-EF >CANT-STOC-N
Da
Nu
VAL-ST-SUPR:= (CANT-STOC-EF – CANT-STOC-N) * PRET-UNITAR
T-VAL-ST-SUPR : = T-VAL-ST-SUPR + VAL-ST-SUPR
VAL-ST-EF: = CANT-STOC-EF * PRET-UNITAR
T-VAL-ST-EF:= T-VAL-ST-EF + VAL-ST-EF
Citește F-CARTELE
Sfîrșit fișier?
Da
Nu
'DA' → SFIRSIT-FISIER

Fig. T.2.4.

```
 ID  DIVISION.
*
 PROGRAM-ID.
     UP01.
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT F-CARTELE   ASSIGN SYSIN.
     SELECT LISTA       ASSIGN SYSOUT.
 DATA DIVISION.
*
 FILE SECTION.
 FD  F-CARTELE        RECORDING F
                      LABEL RECORD OMITTED.
 01  ARTICOL.
   02 GESTIUNE               PIC 9.
     88 GESTIUNE-CORECTA     VALUE 1 THRU 8.
   02 SIMB-MAT               PIC 9(7).
   02 DEN-MAT                PIC X(30).
   02 U-M                    PIC XXX.
   02 PRET-UNITAR            PIC 9(5)V99.
   02 R-PRET-UNITAR REDEFINES PRET-UNITAR PIC X(7).
   02 CANT-STOC-EF           PIC 9(5)V99.
   02 CANT-STOC-N            PIC 9(5)V99.
 FD  LISTA   RECORDING F
             LABEL RECORD OMITTED.
 01  RIND.
   02 FILLER                 PIC X(7).
   02 NRC                    PIC ZZ9.
   02 FILLER                 PIC XXX.
   02 GESTIUNE-L             PIC 9.
   02 FILLER                 PIC XXX.
   02 SIMB-MAT-L             PIC 9(7).
   02 FILLER                 PIC XXX.
   02 DEN-MAT-L              PIC X(30).
   02 FILLER                 PIC XXX.
   02 U-M-L                  PIC XXX.
   02 FILLER                 PIC XXX.
   02 PRET-UNITAR-L          PIC Z(4)9.99.
   02 FILLER                 PIC XXX.
   02 CANT-STOC-EF-L         PIC Z(4)9.99.
   02 FILLER                 PIC XXX.
   02 CANT-STOC-N-L          PIC Z(4)9.99.
 WORKING-STORAGE SECTION.
 77 SFIRSIT-FISIER           PIC XX VALUE 'NU'.
   88 SFIRSIT-FISIER-DA         VALUE 'DA'.
 77  NC                      PIC 999 VALUE 0.
 77  V                       PIC 9    VALUE 0.
 77  T-VAL-ST-EF             PIC 9(12)V99 VALUE 0.
 77  T-VAL-ST-SUPR           PIC 9(7)V99  VALUE 0.
 77  VAL-ST-SUPR             PIC 9(7)V99.
 77  VAL-ST-EF               PIC 9(8)V99.
 PROCEDURE DIVISION.
```

Programul T.2.1.

```
INCEPUT.
    OPEN INPUT F-CARTELE
        OUTPUT LISTA
    READ F-CARTELE AT END MOVE 'DA' TO SFIRSIT-FISIER.
    PERFORM B-LISTARE UNTIL SFIRSIT-FISIER-DA
    DISPLAY 'VALOAREA STOCURILOR EFECTIVE = ' T-VAL-ST-EF
    DISPLAY 'VALOAREA STOCURILOR SUPRANORMATIVE=' T-VAL-ST-SUPR
    CLOSE F-CARTELE
          LISTA
    STOP RUN.
B-LISTARE.
    EXAMINE R-PRET-UNITAR REPLACING LEADING SPACES BY ZEROS
    IF NOT GESTIUNE-CORECTA MOVE 1 TO V
                              DISPLAY ' GESTIUNE VAL.ER. '.
    IF PRET-UNITAR NOT NUMERIC MOVE 1 TO V
                                  DISPLAY 'PRET NENUMERIC'.
    IF V = 1
        THEN
          MOVE 0 TO V
          DISPLAY 'ARTICOL ERONAT = ' ARTICOL
        ELSE
          ADD 1 TO NC
          MOVE SPACES TO RIND
          MOVE NC                TO NRC
          MOVE GESTIUNE          TO GESTIUNE-L
          MOVE SIMB-MAT          TO SIMB-MAT-L
          MOVE DEN-MAT           TO DEN-MAT-L
          MOVE U-M               TO U-M-L
          MOVE PRET-UNITAR       TO PRET-UNITAR-L
          MOVE CANT-STOC-EF      TO CANT-STOC-EF-L
          MOVE CANT-STOC-N       TO CANT-STOC-N-L
          WRITE RIND AFTER 1
     IF CANT-STOC-EF > CANT-STOC-N
          COMPUTE VAL-ST-SUPR = (CANT-STOC-EF -
                                 CANT-STOC-N) *
                                 PRET-UNITAR
          ADD VAL-ST-SUPR TO T-VAL-ST-SUPR.
     MULTIPLY CANT-STOC-EF BY PRET-UNITAR GIVING VAL-ST-EF
          ADD VAL-ST-EF TO T-VAL-ST-EF
          READ F-CARTELE AT END MOVE 'DA' TO SFIRSIT-FISIER.
*
*
```

Programul T.2.1

TESTUL 3 **PROGRAM DE CREARE A UNUI FIȘIER PE BANDA MAGNETICĂ**

Să se elaboreze programul pentru crearea unui fișier pe bandă magnetică, cunoscînd următoarele:

a) fluxul general al datelor (fig. T.3.1):

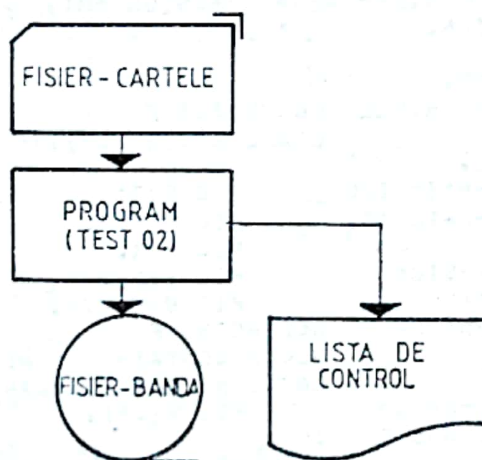


Fig. T.3.1.

b) machetele celor două fișiere (lista de control se va obține folosind instrucțiunea DISPLAY) sînt prezentate în figura T.3.2.

COD MATERIAL	DENUMIRE MATERIAL	UM	CANT STOC	PRET UNITAR
n	a-n	a-n	n 5+2	n 5+2
1 – 7	8 – 37	38 – 41	42 – 48	49 – 55

Pentru fișierul FISIER-BANDA

a-n (55)	Factor de blocaj 50, Etichete standard

Fig. T.3.2.

Observație: Deși în practică operația de memorare a datelor pe suporturi magnetice trebuie să conțină instrucțiuni de validare, pentru mai multă simplitate, acestea vor fi incluse în testul 5.

c) se cere ca la sfîrșit să se imprime numărul de articole înregistrate în fișierul FISIER—BANDA.

Verificați răspunsul cu cel din programul T.3.1.

```
 ID  DIVISION.
**
 PROGRAM-ID.
     TEST03.
*
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT FISIER-CARTELE ASSIGN SYSIN.
     SELECT FISIER-BANDA   ASSIGN AMT.
 DATA DIVISION.
*
 FILE SECTION.
 FD  FISIER-CARTELE   RECORDING F
                      LABEL RECORD OMITTED.
 01  ARTICOL.
     02 COD-MATERIAL        PIC 9(7).
     02 DEN-MATERIAL        PIC X(30).
     02 UM                  PIC XXX.
     02 CANT-STOC           PIC 9(5)V99.
     02 PRET-UNITAR         PIC 9(5)V99.
 FD. FISIER-BANDA     RECORDING F
                      BLOCK CONTAINS 30 RECORDS
                      LABEL RECORD STANDARD.
 01  ARTICOL-BANDA          PIC X(54).
 WORKING-STORAGE SECTION.
 77  IND-SF                 PIC XX VALUE 'NU'.
   88 IND-SF-DA                    VALUE 'DA'.
 77  CONTOR                 PIC 999 VALUE 0.
 PROCEDURE DIVISION.
*
 MODUL-PRINCIPAL.
     OPEN  INPUT FISIER-CARTELE
           OUTPUT FISIER-BANDA
     READ FISIER-CARTELE AT END MOVE 'DA' TO IND-SF.
     PERFORM B-INREGISTRARE UNTIL IND-SF-DA
     DISPLAY SPACES
     DISPLAY ' TOTAL ARTICOLE INREGISTRATE = ' CONTOR
     DISPLAY ' ===================================='
     CLOSE FISIER-CARTELE
           FISIER-BANDA
     STOP RUN.
 B-INREGISTRARE.
     DISPLAY            ARTICOL
     WRITE ARTICOL-BANDA FROM ARTICOL
     ADD 1 TO CONTOR
     READ FISIER-CARTELE AT END MOVE 'DA' TO IND-SF.
*
*
```

Programul T.3.1.

El este asemănător?

DA → Lecția 11

NU → Lecția 9

TESTUL 4

PROGRAM DE SORTARE, FĂRĂ PROCEDURI DE INTRARE ȘI IEȘIRE

Să se elaboreze programul pentru sortarea unui fișier pe bandă magnetică utilizînd sortul COBOL.

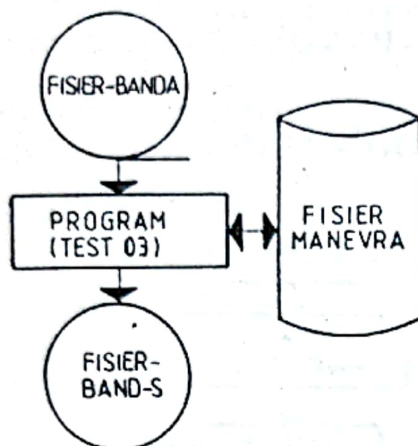


Fig. T.4.1.

Sortarea se va realiza în funcție de:

— valorile crescătoare ale cîmpului de dată COD—MATERIAL.

Structura datelor este cea precizată în testul 3.

Fluxul general al datelor este cel din figura T.4.1.

Verificați structura cu cea prezentată în programul T.4.1.

Ea este asemănătoare?

DA → Lecția 12

NU ↰ 11.7.

```
 ID DIVISION.
*
 PROGRAM-ID.
     TEST04.
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT FISIER-BANDA ASSIGN AMT.
     SELECT FISIER-BANDA-S ASSIGN BMT.
     SELECT FISIER-MANEVRA ASSIGN CZP.
 DATA DIVISION.
*
 FILE SECTION.
 FD  FISIER-BANDA    RECORDING F
                     BLOCK CONTAINS 50 RECORDS
                     LABEL  RECORD STANDARD.
 01  ART-BANDA       PIC X(54).
 FD  FISIER-BANDA-S  RECORDING F
                     BLOCK CONTAINS 50 RECORDS
                     LABEL RECORD STANDARD.
 01  ART-BANDA-S     PIC X(54).
 SD  FISIER-MANEVRA  DATA RECORD ART-MANEVRA.
 01  ART-MANEVRA.
     02 COD-MATERIAL PIC 9(7).
     02 FILLER       PIC X(47).
 PROCEDURE DIVISION.
*
 INCEPUT.
     SORT FISIER-MANEVRA ON ASCENDING KEY COD-MATERIAL
                             USING FISIER-BANDA
                             GIVING FISIER-BANDA-S
     STOP RUN.
*
*
```

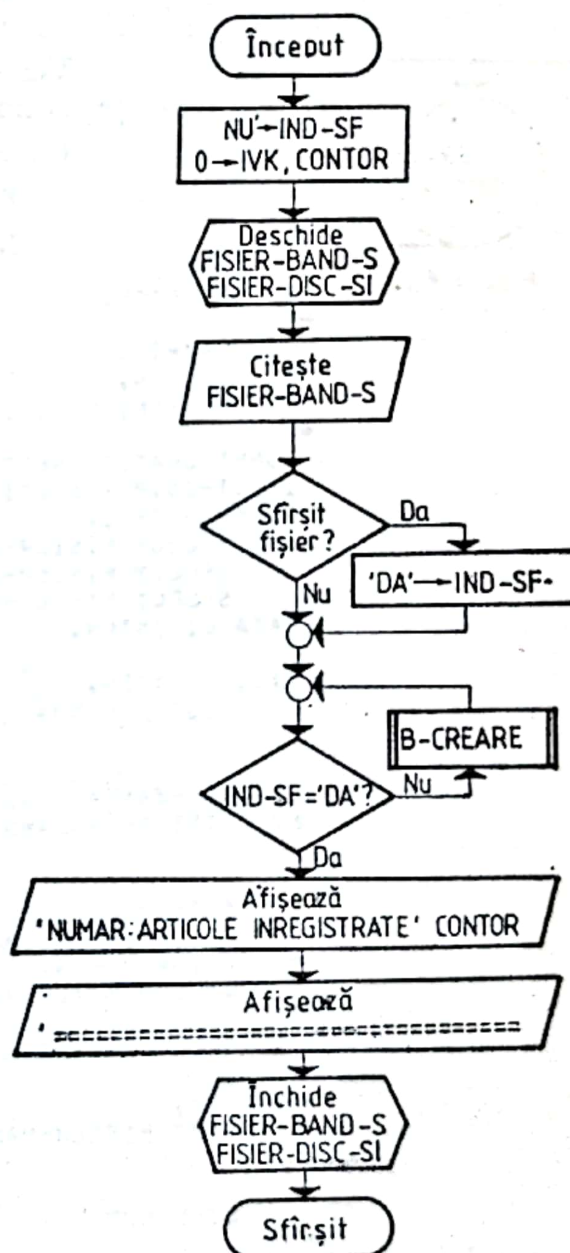
Programul T.4.1.

TESTUL 5 PROGRAM DE CREARE A UNUI FIȘIER CU ORGANIZARE SECVENȚIALĂ — INDEXATĂ

Să se elaboreze programul TESTO5 pentru crearea fișierului FISIER—DISC-SI cu organizarea secvențială-indexată, cunoscînd următoarele:

— în intrare se utilizează fișierul FISIER—BANDĂ—S obținut prin programul din testul 4;

— structura fișierelor este cea din testul 3, cu precizarea că la constituirea tabelelor cu indecși se va utiliza cheia COD—MATERIAL—D (din structura articolului asociat fișierului pe disc);



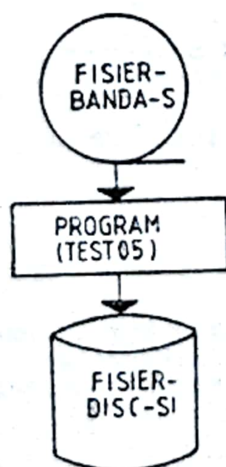


Fig. T.5.1.

Fig. T.5.2.

— fluxul general al datelor este redat în fig. T.5.1.
— cîmpul CANT—STOC—D din structura articolului asociat fișierului pe disc, va fi memorat în format condensat (împachetat).

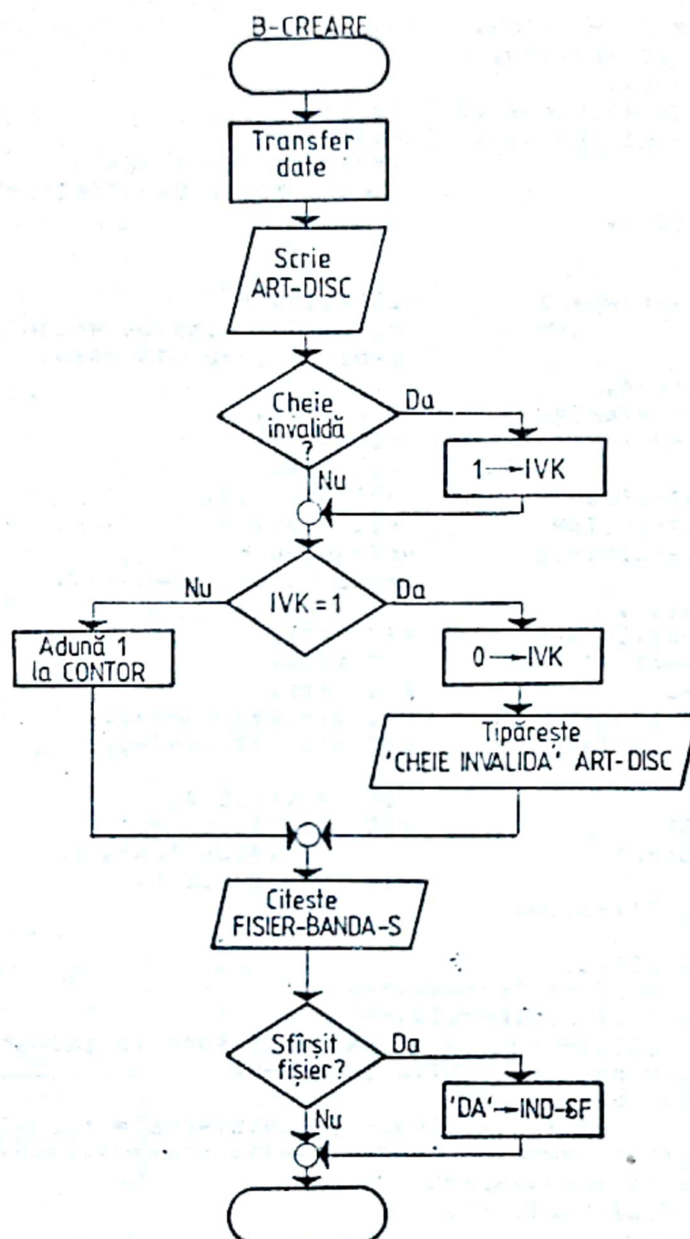


Fig. T.5.3.

Verificați răspunsul cu cel din figurile T.5.2, T.5.3. și Programul T.5.1.

Evidențiați neconcordanțele în denumirea fișierelor din figurilor T.5., T.5.2., T.5.3. Corectați schemele conform identificatorului acordat fișierului pe bandă în programul T.5.1.

El este asemănător?
NU ↰ 12.1.
DA → Testul 6

```
ID DIVISION.
*
PROGRAM-ID.
    TESTU5.
ENVIRONMENT DIVISION.
*
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FISIER-BANDA-S ASSIGN BMT.
    SELECT FISIER-DISC-SI ASSIGN AMD
                          ORGANIZATION INDEXED
                          RECORD KEY COD-MATERIAL-D.
DATA DIVISION.
*
FILE SECTION.
FD  FISIER-BANDA-S        RECORDING F
                           BLOCK CONTAINS 50 RECORDS
                           LABEL RECORD STANDARD.
01  ART-BANDA.
  02  COD-MATERIAL         PIC 9(7).
  02  DEN-MATERIAL         PIC X(30).
  02  UM                   PIC XXX.
  02  CANT-STOC            PIC 9(5)V99.
  02  PRET-UNITAR          PIC 9(5)V99.
FD  FISIER-DISC-SI        RECORDING F
                          LABEL RECORD STANDARD.
01  ART-DISC.
  02 COD-MATERIAL-D       PIC 9(7).
  02 DEN-MATERIAL-D       PIC X(30).
  02  UM-D                PIC  XXX.
  02 CANT-STOC-D          PIC 9(5)V99 COMP-3.
  02 PRET-UNITAR-D        PIC 9(5)V99 COMP-3.
WORKING-STORAGE SECTION.
77  IVK                   PIC  9 VALUE 0.
77  IND-SF                PIC  XX VALUE 'NU'.
 88 IND-SF-DA                    VALUE 'DA'.
77  CONTOR                PIC  999 VALUE 0.
PROCEDURE DIVISION.
*
MODUL-PRINCIPAL.
    OPEN INPUT FISIER-BANDA-S
        OUTPUT FISIER-DISC-SI
    READ FISIER-BANDA-S AT END MOVE 'DA' TO IND-SF.
    PERFORM B-CREARE UNTIL IND-SF-DA
    DISPLAY SPACES
    DISPLAY ' NUMAR ARTICOLE INREGISTRATE = ' CONTOR
    DISPLAY ' ===================================='
    CLOSE FISIER-BANDA-S
          FISIER-DISC-SI
    STOP RUN.
B-CREARE.
    MOVE COD-MATERIAL TO COD-MATERIAL-D
    MOVE DEN-MATERIAL TO DEN-MATERIAL-D
    MOVE UM           TO UM-D
    MOVE CANT-STOC    TO CANT-STOC-D
    MOVE PRET-UNITAR  TO PRET-UNITAR-D
    WRITE ART-DISC INVALID KEY MOVE 1 TO IVK.
    IF IVK = 1 MOVE 0 TO IVK
               DISPLAY ' ART. CU CHEIE ERONATA= ' ART-DISC
         ELSE ADD 1 TO CONTOR.
    READ FISIER-BANDA-S AT END MOVE 'DA' TO IND-SF.
```

Programul T.5.1.

TESTUL 6 PROGRAM DE ACTUALIZARE A UNUI FIŞIER CU ORGANIZARE SECVENŢIALĂ — INDEXATĂ

Să se elaboreze programul pentru actualizarea unui fişier cu organizare secvenţială-indexată, ştiut fiind că actualizarea presupune:

— adăugarea de noi articole;
— modificarea conţinutului unor articole;
— anularea unor articole.

În acest test fluxul general al datelor este prezentat în figura T.6.1.

Pentru a delimita cele trei cazuri, în structura machetei (fig. T.6.2.), s-a prevăzut un cîmp COD—ACTUALIZARE care poate lua valorile *1* pentru adăugări de noi articole, *2* pentru modificări de conţinut şi 3 pentru anulări.

În funcţie de valoarea cîmpului COD—ACTUALIZARE se vor supune prelucrării următoarele tipuri de articole:

COD—ACTUALIZARE = 1
 COD—MATERIAL
 DENUMIRE—MATERIAL
 UM
 CANT—STOC PREŢ-UNITAR
COD—ACTUALIZARE = 2
 COD—MATERIAL
 PREŢ—UNITAR
COD—ACTUALIZARE = 3
 COD—MATERIAL

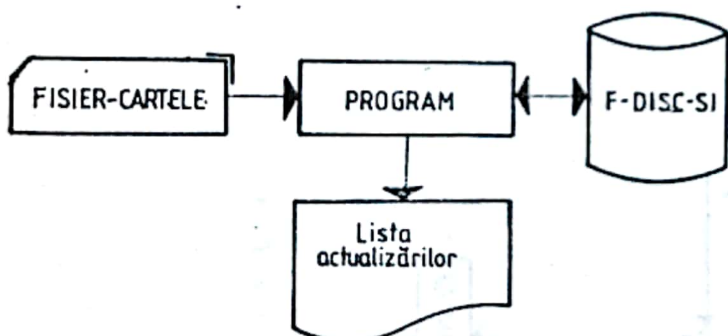


Fig. T.6.1.

COD-ACTUALIZARE	COD-MATERIAL	DENUMIRE-MATERIAL	UM	CANT-STOC	PREŢ-UNITAR
n	n	a-n	a-n	5+2	5+2
1	2 - 8	9 - 38	39 - 41	42 - 48	49 - 56

Fig. T.6.2.

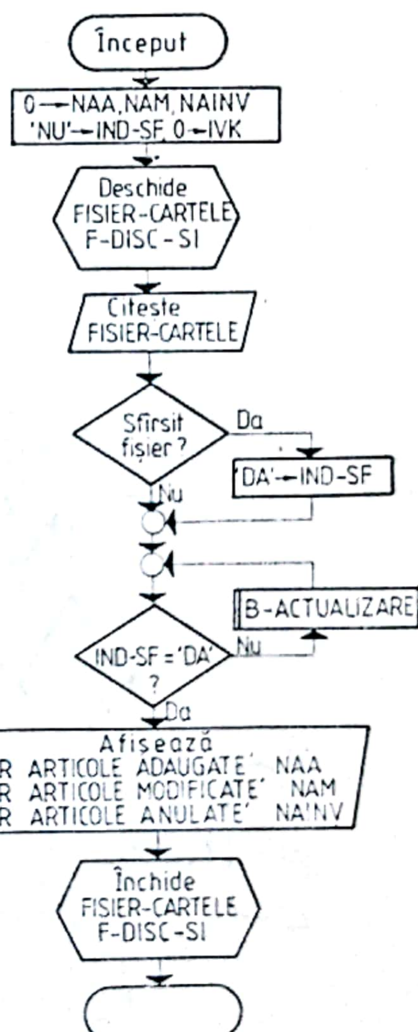


Fig. T.6.3.

B-ACTUALIZARE
COD-ACTUALIZARE = 1
Da
Nu
ADAUGARE
COD-ACTUALIZARE = 2
MODIFICARE
COD-ACTUALIZARE = 3
ANULARE
Afișează ARTICOL ERONAT 'ART-CARTELA
Citește FISIER-CARTELE
Sfîrșit fișier?
'DA'→IND-SF

Fig. T.6.4.

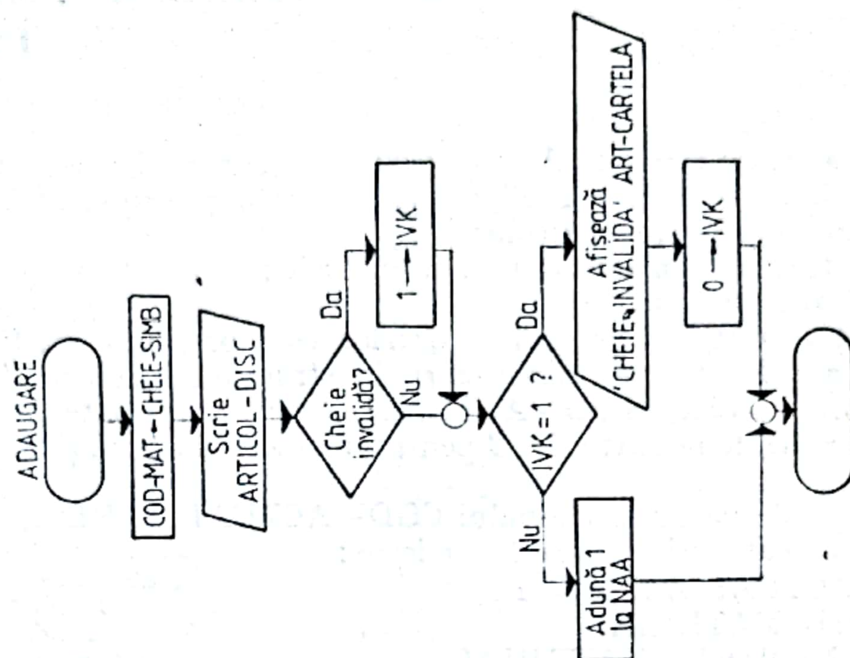


Fig. T.6.5.

Fişierul F—DISC—SI are structura prezentată în testul 5.
Verificaţi răspunsul cu cel din figurile T.6.4., T.6.5., T.6.6. şi programul T.6.1.
El este asemănător?
DA → Lecţia 13
NU ↰ Lecţia 12 paragraful 1

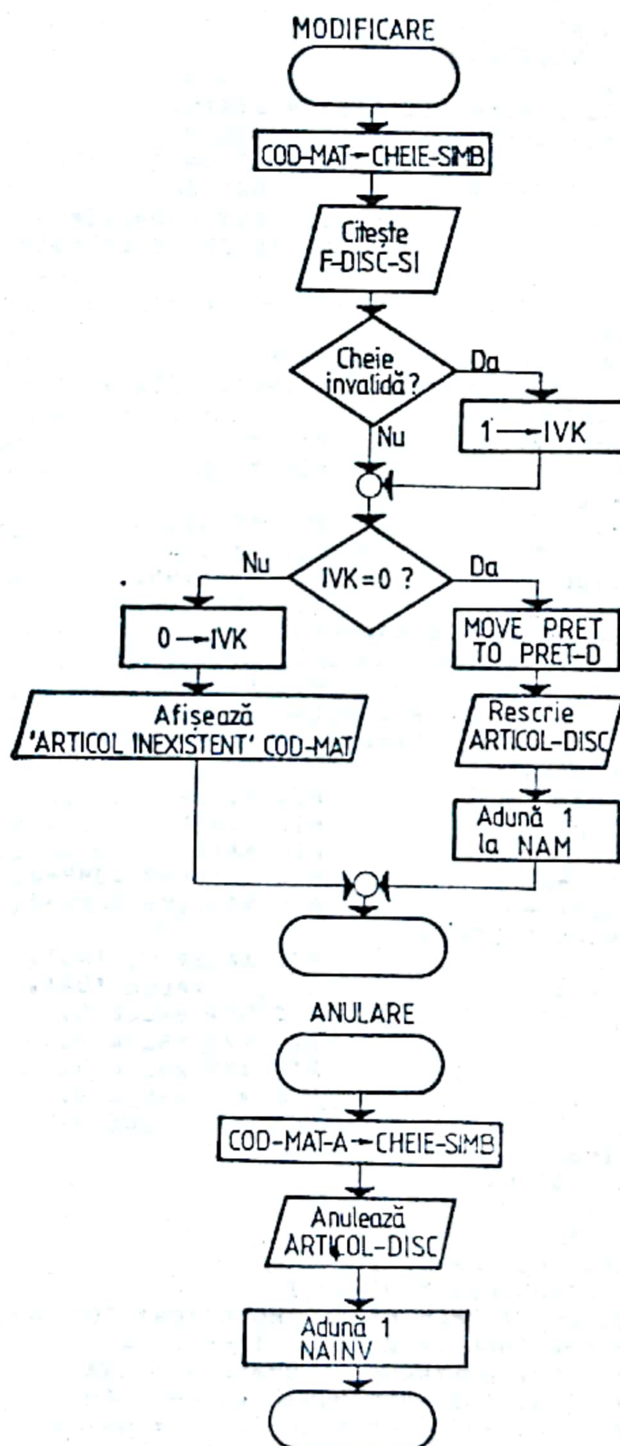


Fig. T.6.6.

```
       ID  DIVISION.
      *
       PROGRAM-ID.
           ACT.
      ** PROGRAMUL *ACT*  REALIZEAZA OPERATIILE DE ACTUALIZARE
      ** A FISIERULUI F-DISC-SI  CU ORGANIZARE SECVENTIALA-INDEXATA
      **
       ENVIRONMENT DIVISION.
      *
       CONFIGURATION SECTION.
       INPUT-OUTPUT  SECTION.
       FILE-CONTROL.
           SELECT FISIER-CARTELE ASSIGN SYSIN.
           SELECT F-DISC-SI      ASSIGN AMD
                                 ORGANIZATION INDEXED
                                 ACCESS RANDOM
                                 RECORD KEY COD-MAT-D
                                 SYMBOLIC KEY CHEIE-SIMB.
       DATA DIVISION.
      *
       FILE SECTION.
       FD  FISIER-CARTELE     RECORDING F
                              LABEL RECORD OMITTED.
       01  ART-CARTELA.
        02 COD-ACTUALIZARE           PIC 9.
         02 COD-MAT                  PIC 9(7).
         02 MACHETA1.
          03 DEN-MAT                 PIC X(30).
          03 UM                      PIC XXX.
          03 CANT-STOC               PIC 9(5)V99.
          03 PRET-UNITAR1            PIC 9(5)V99.
         02 MACHETA2 REDEFINES MACHETA1.
          03 PRET-UNITAR2            PIC 9(5)V99.
          03 FILLER                  PIC X(40).
       FD  F-DISC-SI            RECORDING F
                                LABEL RECORD STANDARD.
       01  ARTICOL-DISC.
         02 COD-MAT-D                PIC 9(7).
         02 DEN-MAT-D                PIC X(30).
         02 UM-D                     PIC XXX.
         02 CANT-STOC-B              PIC 9(5)V99 COMP-3.
         02 PRET-UNITAR-D            PIC 9(5)V99 COMP-3.
       WORKING-STORAGE SECTION.
       77  IND-SF                    PIC XX VALUE 'NU'.
          88 IND-SF-DA                      VALUE 'DA'.
       77  NAA                       PIC 999 VALUE 0.
       77  NAM                       PIC 999 VALUE 0.
       77  NAINV                     PIC 999 VALUE 0.
       77  IVK                       PIC 9   VALUE 0.
         88 IVK-DA                           VALUE 1.
       77  CHEIE-SIMB                PIC 9(7).
       PROCEDURE DIVISION.
      *
       MODUL-PRINCIPAL.
           OPEN INPUT FISIER-CARTELE
                INPUT-OUTPUT F-DISC-SI
           READ FISIER-CARTELE AT END MOVE 'DA' TO IND-SF.
           PERFORM B-ACTUALIZARE UNTIL IND-SF-DA
           DISPLAY ' NR. ARTICOLE ADAUGATE = ' NAA
           DISPLAY ' NR. ARTICOLE MODIFICATE=' NAM
           DISPLAY ' NR. ARTICOLE ANULATE   = ' NAINV
           CLOSE FISIER-CARTELE
                 F-DISC-SI
```

Programul T.6.1.

```
        STOP RUN.
    B-ACTUALIZARE.
        IF COD-ACTUALIZARE = 1
           THEN
              PERFORM ADAUGARE
           ELSE
              IF COD-ACTUALIZARE = 2
                 THEN
                   PERFORM MODIFICARE
                 ELSE
                    IF COD-ACTUALIZARE = 3
                       THEN
                          PERFORM ANULARE
                       ELSE
                          DISPLAY ' ARTICOL ERONAT =' ART-CARTELA.
        READ FISIER-CARTELE AT END MOVE 'DA' TO IND-SF.
    ADAUGARE.
        MOVE COD-MAT TO CHEIE-SIMB
        WRITE ARTICOL-DISC INVALID KEY MOVE 1 TO IVK.
        IF IVK = 1
           THEN
              DISPLAY ' CHEIE INV. ' ART-CARTELA
              MOVE 0 TO IVK
           ELSE
              MOVE 1 TO NAA.
    MODIFICARE.
        MOVE COD-MAT TO CHEIE-SIMB
        READ F-DISC-SI INVALID KEY MOVE 1 TO IVK.
        IF IVK = 1
           THEN
              MOVE 0 TO IVK
              DISPLAY ' ARTICOL INEXISTENT=' COD-MAT
           ELSE MOVE PRET-UNITAR2 TO PRET-UNITAR-D
                REWRITE ARTICOL-DISC
                ADD 1 TO NAM.
    ANULARE.
        MOVE COD-MAT TO CHEIE-SIMB
        DELETE ARTICOL-DISC
        ADD 1 TO NAINV.
   *
   **
```

Programul T.6.1. (continuare)

Î. Ați observat neconcordanțele dintre numerele programelor din figuri și din liste pentru testele 3, 4?

R. Numele programelor din figuri au ultimul caracter cu o unitate mai mică decît cele din liste. Ex: TEST 02 în loc de TEST 03.

TESTUL 7 PROGRAM DE CONTROL DATE ȘI CREARE FIȘIER CU ARTICOLE CORECTE

Să se elaboreze programul pentru controlul datelor și crearea unui fișier (pe bandă magnetică) cunoscînd următoarele:
— fluxul general al datelor este cel din figura T.7.1;

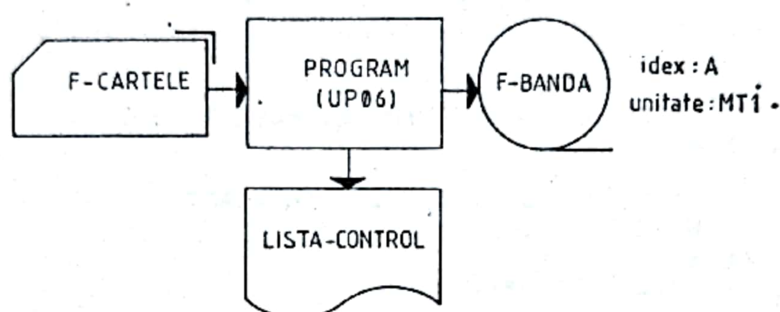


Fig. T.7.1.

— structura articolelor corespunzătoare celor trei fișiere este:
— pentru F—CARTELE (fig. T.7.2);

GESTIUNE	DOCUMENT				SIMBOL-MATERIAL	CANTITATE INTRATĂ	
	NR.	DATA					
		ZI	LUNĂ	AN			
n 2	n 5	n 2	n 2	n 2	n 7	n 5+2	

Fig. T.7.2.

— pentru F—BANDA (fig. T.7.3)
— pentru fișierul LISTA—CONTROL, articolul are aceeași structură ca cea definită la fișierul pe cartele cu mențiunea că între date se lasă cîte

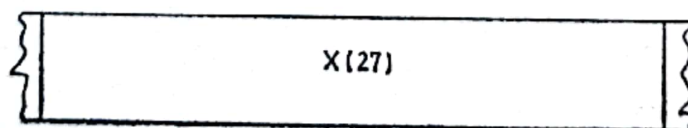


Fig. T.7.3.

4 spații. Articolele eronate vor fi precedate în situație de mesaje indicînd natura erorilor detectate. Într-o reprezentare simplificată cerințele problemei sînt redate în fig. T.7.4.

Operaţiile de control se referă la cîmpurile:

- ZI, pentru a vedea dacă valorile sînt de natură numerică şi cuprinse între 01 şi 31;
- LUNA, pentru a vedea dacă valorile sînt de natură numerică şi cuprinse între 01 şi 12
- CANTITATE—INTRATA, pentru a vedea dacă valorile sînt de natură numerică şi mai mari ca zero.

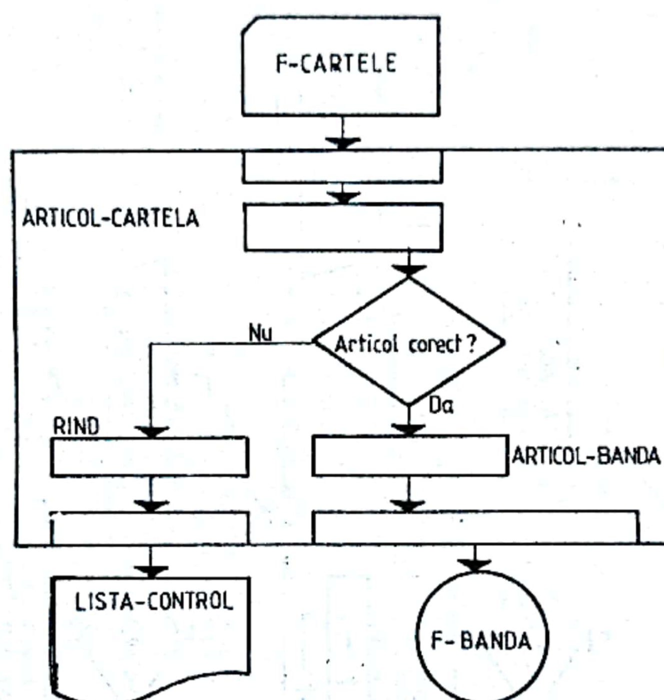


Fig. T.7.4.

Se vor utiliza trei contoare şi anume:

- CONTOR—1, pentru a calcula numărul de articole corecte;
- CONTOR—2, pentru a calcula numărul de articole eronate;
- CONTOR—3, pentru a calcula numărul total de articole.

În schema afişarea conţinutului celor trei memorii (CONTOR—1, CONTOR—2 şi CONTOR 3) s-a precizat prin operaţia „afişează CONTOR".

Schema logică de program este prezentată în fig. T.7.5.

Scrieţi programul pe formularul de programare COBOL. El are o structură asemănătoare cu Programul T.7.1.?

NU ↰ Lecţia 12
DA → Testul 8

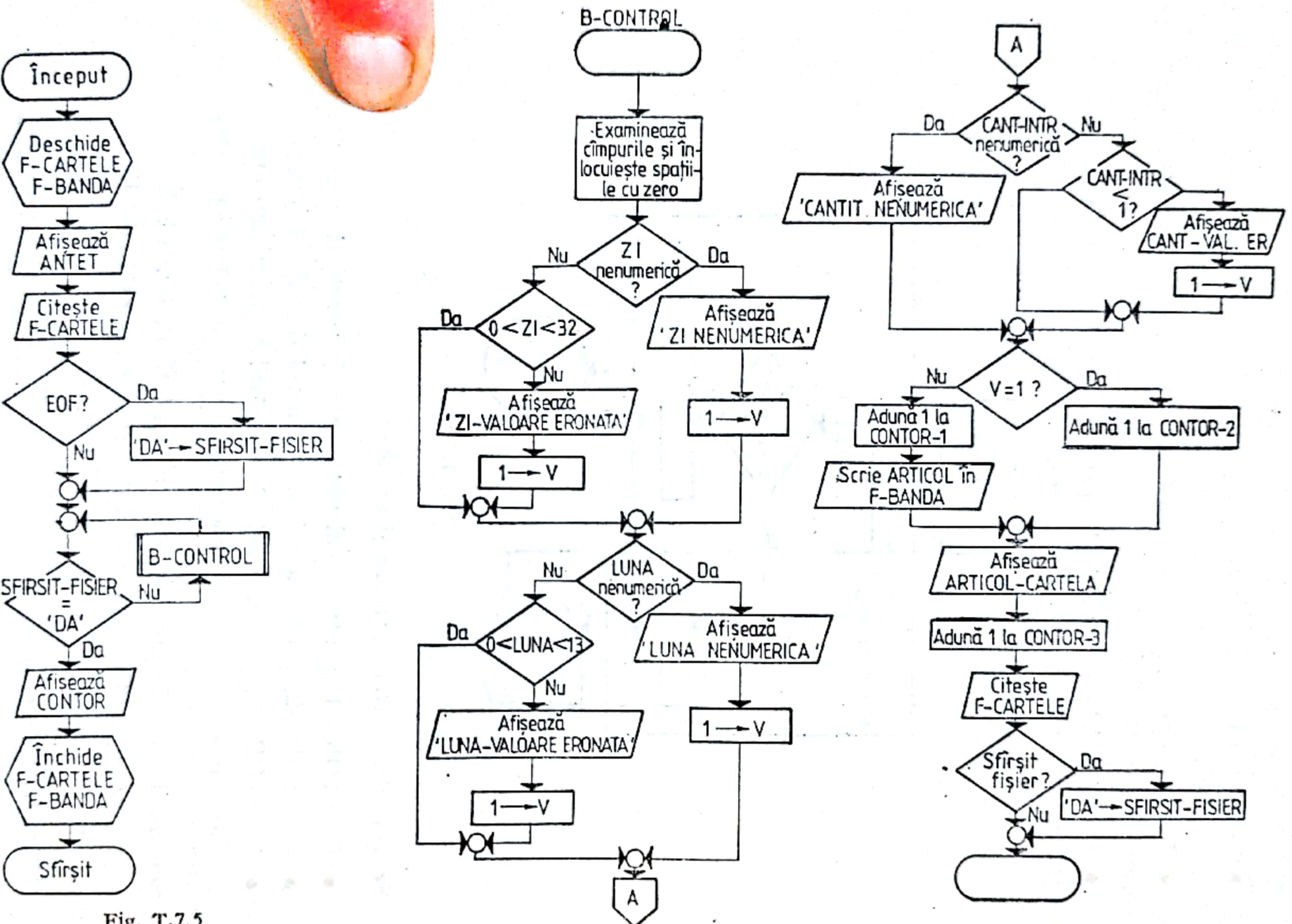

Început
Deschide F-CARTELE F-BANDA
Afișează ANTET
Citește F-CARTELE
EOF?
Da
Nu
'DA'→SFIRSIT-FISIER
SFIRSIT-FISIER = 'DA'
B-CONTROL
Afișează CONTOR
Închide F-CARTELE F-BANDA
Sfîrșit
B-CONTROL
Examinează cîmpurile și înlocuiește spațiile cu zero
ZI nenumerică ?
Afișează 'ZI NENUMERICA'
1→V
0<ZI<32
Afișează 'ZI-VALOARE ERONATA'
LUNA nenumerică ?
Afișează 'LUNA NENUMERICA'
0<LUNA<13
Afișează 'LUNA-VALOARE ERONATA'
A
CANT-INTR nenumerică ?
Afișează 'CANTIT NENUMERICA'
CANT-INTR < 1?
Afișează CANT-VAL. ER
V=1 ?
Adună 1 la CONTOR-1
Scrie ARTICOL în F-BANDA
Adună 1 la CONTOR-2
Afișează ARTICOL-CARTELA
Adună 1 la CONTOR-3
Citește F-CARTELE
Sfîrșit fișier ?
'DA'→SFIRSIT-FISIER

Fig. T.7.5.

```
ID DIVISION.
*
PROGRAM-ID.
    UP06.
ENVIRONMENT DIVISION.
*
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT F-CARTELE ASSIGN SYSIN.
    SELECT  F-BANDA   ASSIGN AMT.
DATA DIVISION.
*
FILE SECTION.
FD  F-CARTELE  RECORDING F
              LABEL RECORD OMITTED.
01  ARTICOL-CARTELA.
 02 GESTIUNE                PIC 9.
 02 DOCUMENT.
  03 NUMAR                  PIC 9(5).
  03 DATA-OP.
   04 ZI                    PIC 99.
   04 RZI REDEFINES ZI      PIC XX.
   04 LUNA                  PIC 99.
   04 RLUNA REDEFINES LUNA    PIC XX.
   04 AN                    PIC 99.
 02 SIMBOL-MATERIAL         PIC 9(7).
 02 CANTITATE-INTR          PIC 9(5)V99.
 02 RCANTITATE-INTR REDEFINES CANTITATE-INTR PIC X(7).
FD  F-BANDA  RECORDING F
             BLOCK CONTAINS 20 RECORDS
             LABEL RECORD STANDARD.
01  ARTICOL-BANDA           PIC X(26).
WORKING-STORAGE SECTION.
77  SFIRSIT-FISIER           PIC XX VALUE  'NU'.
 88 SFIRSIT-FISIER-DA               VALUE  'DA'.
77  V                        PIC 9  VALUE 0.
77  CONTOR-1                 PIC 999 VALUE 0.
77  CONTOR-2                 PIC 999 VALUE 0.
77  CONTOR-3                 PIC 999 VALUE 0.
PROCEDURE DIVISION.
INCEPUT.
    OPEN INPUT F-CARTELE
        OUTPUT F-BANDA
    DISPLAY SPACES
    DISPLAY '    *** L I S T A  D E  C O N T R O L ***    '
    DISPLAY '    ====================================    '
    DISPLAY SPACES
    READ F-CARTELE AT END MOVE 'DA' TO SFIRSIT-FISIER.
    PERFORM B-CONTROL UNTIL  SFIRSIT-FISIER-DA
    DISPLAY SPACES
    DISPLAY ' AU FOST CITITE = ' CONTOR-3 ' ARTICOLE '
    DISPLAY '  DIN CARE CORECTE = ' CONTOR-1  ' ARTICOLE'
    DISPLAY '           ERONATE = ' CONTOR-2  ' ARTICOLE'
    CLOSE F-CARTELE
          F-BANDA
```

Programul T.7.1.

```
        STOP RUN.
    B-CONTROL.
        EXAMINE RZI                 REPLACING LEADING SPACES BY ZEROS
        EXAMINE RLUNA               REPLACING LEADING SPACES BY ZEROS
        EXAMINE RCANTITATE-INTR     REPLACING LEADING SPACES BY ZEROS
        IF ZI NOT NUMERIC  MOVE  1 TO V
                           DISPLAY 'ZI NENUMERICA '
            ELSE IF ZI = 0 OR ZI > 31  MOVE 1 TO V
                                    DISPLAY 'ZI VALOARE ERONATA ',
        IF LUNA NOT NUMERIC MOVE 1 TO V
                            DISPLAY 'LUNA NENUMERICA '
              ELSE IF LUNA = 0   OR LUNA > 12
                                         MOVE 1 TO V
                                         DISPLAY 'LUNA VALOARE ER.'.
        IF CANTITATE-INTR NOT NUMERIC MOVE 1 TO V
                                      DISPLAY ' CANTITATE NENUM.'
            ELSE IF CANTITATE-INTR = 0 MOVE 1 TO V
                                      DISPLAY ' CANT. VALOARE ER.'.
        IF  V = 0
              THEN
                  WRITE ARTICOL-BANDA FROM ARTICOL-CARTELA
                  ADD 1 TO CONTOR-1
              ELSE
                  ADD 1 TO CONTOR-2
                  MOVE 0 TO V.
        DISPLAY ARTICOL-CARTELA
        ADD 1 TO CONTOR-3
        READ F-CARTELE AT END MOVE 'DA' TO SFIRSIT-FISIER.
   *
   *
```

Programul 7.1. (continuare)

TESTUL 8 PROGRAM PENTRU IMPRIMAREA UNUI RAPORT

Să se elaboreze programul COBOL pentru obținerea listei ,,Situația intrărilor de materiale pe feluri de materiale și gestiuni'' folosind în intrare fișierele F—BANDA (cu organizare secvențială) și F—DISC—SI (cu organizare secvențială-indexată). Conținutul celor două fișiere a fost prezentat în testele precedente.

Întrucît articolele din F—BANDA trebuie sortate în funcție de valorile crescătoare ale cîmpurilor GESTIUNE și SIMBOL—MATERIAL se sugerează folosirea instrucțiunii SORT în varianta cu procedură de ieșire.

Fluxul general al datelor este cel din figura T.8.1.

Intrările de materiale vor fi înregistrate în fișierul stocurilor în vederea realizării operațiilor de actualizare. Se va face totodată testarea pentru a detecta materialele cu stocuri supranormative.

Structura articolelor corespunzătoare fișierelor este următoarea :

— pentru fișierul F—BANDA cea prezentată în testul 7;
— pentru fișierul F—DISC—SI cea prezentată în testele 5 și respectiv 6;

— pentru fișierul de manevră se asociază aceeași structură ca la fișierul F—BANDA;

— pentru fișierul SITUATIE articolul descris în secțiunea FILE are lungimea de 100 caractere alfanumerice.

Remarcă

Structura fiecărui rînd (fig; T.8.2.) va fi descrisă în secțiunea memorie de lucru (cu excepția rîndurilor R1—R5 care vor fi imprimate folosind instrucțiunea DISPLAY). Programul obținut este asemănător cu programul 16.2.?

Deși la compilare programul nu prezintă erori de sintaxă pentru lansare în execuție marcați caracterul punct în B—PRELUCRARE după WRITE RIND FROM R8 AFTER1.

DA → Lecția 17.
NU → 14.1

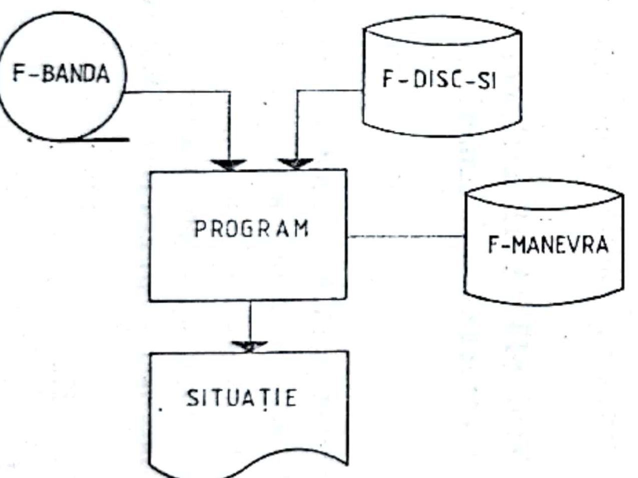


Fig. T.8.1.

R1 ÎNTREPRINDEREA: xxxxxxxxxxx PAG: ZZ9
R2 DATA: 99/99/1999
R3 SITUAȚIA INTRĂRILOR DE MATERIALE PE FELURI DE MAT. ȘI GESTIUNI
R4
R5 GESTIUNEA: 9
R6
R7

	NR-CRT	FEL-MAT	DENUMIRE	UM	CANT-INTR	PRET-UNITAR	VALOARE
R6							
R8	ZZ9	9(7)	X(30)	xxx	Z(4)9.99	Z(4)9.99	Z(8)9.99
R9			TOTAL FEL-MATERIAL		Z(6)9.99		Z(9)9.99
R10	TOTAL GESTIUNE						Z(10)9.99

Fig. T.8.2.

```
 ID DIVISION.
*
 PROGRAM-ID.
     UP07.
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT F-DISC-SI  ASSIGN AMD
                       ORGANIZATION INDEXED
                       ACCESS RANDOM
                       RECORD KEY SIMBOL-MATERIAL-D
                       SYMBOLIC KEY SIMBOL-MATERIAL-W.
     SELECT SITUATIE ASSIGN SYSOUT.
     SELECT F-BANDA ASSIGN BMT.
     SELECT F-MANEVRA ASSIGN CAD.
 DATA DIVISION.
*
 FILE SECTION.
 FD   F-DISC-SI    RECORDING F
                   LABEL RECORD STANDARD
                   BLOCK CONTAINS 30 RECORDS.
 01   ARTICOL-DISC.
   02 GESTIUNE-D               PIC 9.
   02 SIMBOL-MATERIAL-D        PIC 9(7).
   02 DEN-MATERIAL-D           PIC X(30).
   02 U-M-D                    PIC XXX.
   02 PRET-UNITAR-D            PIC 9(5)V99.
   02 CANT-STOC-EF-D           PIC 9(5)V99.
   02 CANT-STOC-N-D            PIC 9(5)V99.
 FD   SITUATIE   RECORDING F
                 LABEL RECORD OMITTED.
 01   RIND                     PIC X(100).
 FD   F-BANDA             RECORDING F
                          BLOCK CONTAINS 20 RECORDS
                          LABEL RECORD STANDARD.
 01   ARTICOL-BANDA            PIC X(26).
 SD   F-MANEVRA           DATA RECORD ARTICOL.
 01   ARTICOL.
   02 GESTIUNE                 PIC 9.
   02 DOCUMENT.
    03 NUMAR                   PIC 9(5).
    03 DATA-OP.
     04 ZI                     PIC 99.
     04 LUNA                   PIC 99.
     04 AN                     PIC 99.
   02 SIMBOL-M                 PIC 9(7).
   02 CANT-INT                 PIC 9(5)V99.
 WORKING-STORAGE SECTION.
 77   C-PAG                    PIC 99 VALUE 0.
 77   SIMBOL-MATERIAL-W        PIC 9(7).
 77   IVK                      PIC 9 VALUE 0.
  88  IVK-DA                         VALUE 1.
 77   IND-SF                   PIC 9 VALUE 0.
  88  IND-SF-DA                      VALUE 1.
 01   DATA-SISTEM.
   02 AN1                      PIC 99.
   02 LUNA1                    PIC 99.
   02 ZI1                      PIC 99.
```

Programul T.8.1.

```
01  IND-GESTIUNE.
  02 Z-GESTIUNE                  PIC 9.
  02  IND-SF-G                   PIC 9 VALUE 0.
   88 IND-SF-G-DA                      VALUE 1.
  02 TOTAL-GEST                  PIC 9(11)V99 VALUE 0.
01  IND-FEL-MAT.
  02 Z-MATERIAL                  PIC 9(7).
  02 IND-SF-M                    PIC 9 VALUE 0.
   88 IND-SF-M-DA                       VALUE 1.
  02  TOTAL-CFM                  PIC 9(7)V99 VALUE 0.
  02  TOTAL-VFM                  PIC 9(10)V99 VALUE 0.
01  DATE-INDEP.
  02  VALOARE                    PIC 9(9)V99.
  02  NC                         PIC 999 VALUE 0.
  02  CRIND                       PIC 99   VALUE 62.
  02  DENUMIRE-INTREPRINDERE PIC  X(11).
01  R6.
 02 FILLER                       PIC X(6)  VALUE SPACES.
 02 FILLER                       PIC X(86)  VALUE ALL '='.
01  R7.
 02 FILLER                       PIC X(6)  VALUE SPACES.
 02 FILLER   PIC X(86)  VALUE ' NR-CRT = SIMN-MAT = DENUMIRE
-     '   =  UM = CANT-INTR = PRET-UNITAR = VALOARE =      '.
01  R8.
 02 FILLER   PIC X(6)  VALUE SPACES.
 02 NRC      PIC  ZZ9.
 02 FILLER   PIC  XXX VALUE SPACES.
 02 SIMBOL-MAT-L PIC 9(7).
 02 FILLER   PIC  XXX VALUE SPACES.
 02 DENUMIRE-L PIC X(30).
 02 FILLER   PIC  XXX VALUE SPACES.
 02 UM-L     PIC  XXX.
 02 CANT-INTR-L  PIC BBBZ(4)9.99.
 02 PRET-UNITAR-L      PIC BBBZ(4)9.99.
 02 VALOARE-L          PIC BBBZ(8)9.99.
01  R9.
   02 FILLER           PIC X(20)  VALUE SPACES.
   02 FILLER   PIC X(21)  VALUE ' TOTAL FEL MATERIAL ='.
   02 TOTAL-CFM-L      PIC Z(6)9.99.
   02 TOTAL-VFM-L      PIC BBBBZ(9)9.99.
01  R10.
   02  FILLER  PIC X(20)  VALUE SPACES.
  02  FILLER  PIC X(21)  VALUE ' TOTAL GESTIUNE    ='.
   02  TOTAL-GEST-L  PIC Z(10)9.99.
PROCEDURE DIVISION.
*
 SORTARE SECTION.
 INCEPUT.
     SORT F-MANEVRA ASCENDING GESTIUNE
                              SIMBOL-M
                    USING F-BANDA
                    OUTPUT PROCEDURE PROCEDURA-LISTARE
     STOP RUN.
 PROCEDURA-LISTARE SECTION.
 P1.
     ACCEPT DATA-SISTEM FROM DATE
     ACCEPT DENUMIRE-INTREPRINDERE
     OPEN INPUT-OUTPUT F-DISC-SI
         OUTPUT        SITUATIE
     RETURN F-MANEVRA AT END MOVE 1 TO IND-SF.
     PERFORM B-GESTIUNE UNTIL IND-SF-DA
     CLOSE F-DISC-SI
          SITUATIE
     GO TO SF.
```

```
B-GESTIUNE.
    MOVE GESTIUNE         TO Z-GESTIUNE
    MOVE 0                TO IND-SF-G
    MOVE 0                TO TOTAL-GEST
    PERFORM B-SIMBOL UNTIL IND-SF-G-DA
    MOVE TOTAL-GEST TO TOTAL-GEST-L
    WRITE RIND FROM R10 AFTER 2
    MOVE 63 TO CRIND.
B-SIMBOL.
    MOVE SIMBOL-M         TO SIMBOL-MATERIAL-W
                             Z-MATERIAL
    MOVE 0                TO IND-SF-M
                             IVK
                             TOTAL-CFM
                             TOTAL-VFM
    READ F-DISC-SI INVALID KEY MOVE 1 TO IVK.
    PERFORM B-PRELUCRARE UNTIL IND-SF-M-DA
    IF IVK-DA NEXT SENTENCE
       ELSE
           ADD CANT-INT TO CANT-STOC-EF-D
           REWRITE ARTICOL-DISC.
    PERFORM FIXARE-PAGINA
    MOVE TOTAL-CFM        TO TOTAL-CFM-L
    MOVE TOTAL-VFM        TO TOTAL-VFM-L
    WRITE RIND FROM R9 AFTER 1
    ADD TOTAL-VFM TO TOTAL-GEST.
B-PRELUCRARE.
    IF IVK-DA
      THEN
        DISPLAY 'LIPSA DATE PERMANENTE = ' SIMBOL-M
      ELSE
        MULTIPLY CANT-INT BY PRET-UNITAR-D GIVING VALOARE
        ADD CANT-INT            TO TOTAL-CFM
        ADD VALOARE             TO TOTAL-VFM
        PERFORM FIXARE-PAGINA
        ADD 1 TO NC
        MOVE NC                 TO NRC
        MOVE SIMBOL-M           TO SIMBOL-MAT-L
        MOVE DEN-MATERIAL-D     TO DENUMIRE-L
        MOVE U-M-D              TO UM-L
        MOVE CANT-INT           TO CANT-INTR-L
        MOVE PRET-UNITAR-D      TO PRET-UNITAR-L
        MOVE VALOARE            TO VALOARE-L
        WRITE RIND FROM R8 AFTER 1
        RETURN F-MANEVRA AT END MOVE 1 TO IND-SF
                                          IND-SF-M
                                          IND-SF-G.
        IF IND-SF = 0
            IF GESTIUNE NOT = Z-GESTIUNE
                            MOVE 1 TO IND-SF-M
                                      IND-SF-G
               ELSE IF SIMBOL-M NOT = Z-MATERIAL
                            MOVE 1 TO IND-SF-M.
FIXARE-PAGINA.
    ADD 1 TO CRIND
    IF CRIND > 60 MOVE SPACES TO RIND
                  WRITE RIND AFTER 0
                  ADD 1 TO C-PAG
                  DISPLAY SPACES
            DISPLAY ' INTREPRINDEREA = 'DENUMIRE-INTREPRINDERE
            DISPLAY ' DATA =' DATA-SISTEM  'PAG=' C-PAG
            DISPLAY ' SITUATIA INTRARILOR DE MAYERIALE PE FELURI
-    ' DE MATERIALE SI GESTIUNI '
            DISPLAY ' ==========================================
-    '========================= '
            DISPLAY 'GESTIUNEA: ' Z-GESTIUNE
            WRITE RIND FROM R6 AFTER 1
            WRITE RIND FROM R7 AFTER 1
            WRITE RIND FROM R6 AFTER 1
            MOVE 0 TO CRIND.
SF.
    EXIT.
*
*
```

BIBLIOGRAFIE

1. * * * Hotărîrea C.C. al P.C.R. privind perfecționarea sistemului informațional economic-social, introducerea sistemelor de conducere uu mijloace de prelucrare automată a datelor și dotarea economiei naționale cu tehnică de calcul în perioada 1971—1980, Scînteia, 22.IX.1972
2. **Nicolae Ceaușescu** Raport la cel de al XII-lea Congres al P.C.R., Editura politică
3. **Amstrong, R. M.** Modular Programming in COBOL, John Wiley and Sons, 1973
4. **Baltac, V.** Felix C—256. Structura și programarea calculatorului, Editura tehnică, București, 1974
5. **Bertini, M. T. Tallineau I.** Le COBOL structurée, Ed. d'nformatique, Paris, 1975
6. **Dahl, O. J., Dijkstra, E. W. Hoare, C.A.R.** Structured Programming, Academic Press, London, 1972
7. **Dykstra, E. H.** A Discipline of Programming, Englewood Cliffs, New Jersey, Prentice all, Inc., 1976
8. **Drăghici, M.** Inițiere în COBOL, București, Editura tehnică, 1972
9. **Farcaș, D.** Calculatorul electronic și gîndirea umană, Editura Albatros, București, 1979
10. **Farcaș, D. Mihoci, N.** Diferențe ale limbajului COBOL IRIS—50 față de limbajul COBOL DOS IGM/360 Rev. CEPECA nr. 3/1972
11. **Myers, G. I.** Reliable Software through Composite Design, New-York, Charter Publishers, 1975
12. **Niculescu, St.** FORTRAN. Inițiere în programarea structurată, București, Editura Tehnică, 1979
13. **Văduva, I. Baltac, V. Florescu, V. Floricică, I.** Programare structurată, Editura tehnică, 1978
14. **Văduva, I. Popa, Gh. Florescu, V.** Programarea calculatoarelor electronice prin metoda instruirii programate, ASE, uz intern 1979
15. **Văduva, I. Popa, Gh. Florescu, V.** Culegere de probleme, pentru partea a II-a „Limbaje de programare a calculatoarelor electronice" a cursului „Prelucrarea electronică a informației financiar-contabile", ASE, București 1977
16. **Văduva, I. Popa Gh. Florescu, V.** Culegere de probleme. Scheme logice, ASE, București 1979
17. * * * La programmation structurée. Infotech State of The Art Raport (England) et Edition d'Informatique (France). Traduit de l'anglais, 1977



ANEXA 1. Formate generale COBOL, cuvinte rezervate, cod EBCDIC

Părţile unui program COBOL
1. IDENTIFICATION DIVISION.
2. ENVIRONMENT DIVISION.
3. DATA DIVISION.
4. PROCEDURE DIVISION.

```
8
{ IDENTIFICATION DIVISION. }
{ ID DIVISION.             }

  PROGRAM-ID. { nume-program.   }
              { 'nume-program'. }

[AUTHOR. nume-autor.]
[INSTALLATION. paragraf-comentariu.]
[DATE-WRITTEN. paragraf-comentariu.]
[DATE-COMPILED. paragraf-comentariu.]
[SECURITY. paragraf-comentariu.]
[REMARKS. paragraf-comentariu.]

8    12
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
[SOURCE-COMPUTER. paragraf-comentariu.]
[OBJECT-COMPUTER. SEGMENT-LIMIT IS număr-prioritate]
                  [linii-comentariu].]
[SPECIAL-NAMES. [CURRENCY SIGN IS literal]
                [DECIMAL-POINT IS COMMA]

                [ { CONSOLE
                  { { SYSIN  }
                  { { SYSIPT }
                  { { SYSOUT }
                  { { SYSLST }
                  { { SYSOPT }
                  { { SYSPUNCH } } IS nume-mnemonic]...].
                  { { SYSPCH   }
                  {   CØ1
                  {   CØ2
                  {   CØ3
                  {   CØ4
                  {   CØ5
                  {   CØ6
                  {   CØ7
                  {   CØ8
                  {   CSP
```

```
8     12
[INPUT—OUTPUT SECTION].
 FILE—CONTROL.
          {SELECT [OPTIONAL] nume—de—fişier

                    ⎧ idex—1
                    ⎪ idex—2
                    ⎪ {SYSIN
                    ⎪  SYSIPT}
                    ⎪ {SYSOUT
     ASSIGN TO      ⎨  SYSLST
                    ⎪  SYSOPT}
                    ⎪ {SYSPUNCH
                    ⎪  SYSPCH}
                    ⎪ SYSRERUN
                    ⎪ SYSMAN
                    ⎩ [întreg—1] idex—a [idex—b] ...

   [FOR MULTIPLE {REEL
                  UNIT}]

   [RESERVE   {întreg—1
               NO}]   ALTERNATE {AREAS
                                 AREA}]

   [ACCESS MODE IS {SEQUENTIAL
                    RANDOM}]

   [ORGANIZATION MODE IS {SEQUENTIAL
                          INDEXED
                          DIRECT}].

8     12
  [PROCESSING MODE IS SEQUENTIAL
  [{FILE—LIMIT IS     }{nume—dată—1}  THRU  {nume—dată—2}
   {FILE—LIMITS ARE   }{literal—1  }        {literal—2  }
  [{nume—dată—3} THRU {nume—dată—4   } ...]
   {literal—3  }      {literal—4...  }
```

```
        [ACTUAL KEY IS nume-de-dată-1]
        [RECORD KEY IS nume-de-dată-2]
        [SYMBOLIC KEY IS nume-de-dată-3].} ...
[ I-O-CONTROL.

  [APPLY LOCATE MODE ON nume-fişier ...]

  [                                      {       {REEL}  }  ]
  [ RERUN [ON nume-fişier] EVERY         { END OF {UNIT}  }  ]
  [                                      { întreg RECORDS }  ]
  [                                        OF nume-fişier    ] ...
  [SAME [{RECORD}] AREA FOR nume-fişier ...                  ] ...
        [{SORT  }]
  [MULTIPLE FILE TAPE CONTAINS
                      nume -fişier [POSITION întreg]
                      [nume-fişier [POSITION întreg] ]... ] ... ]

8   12
DATA DIVISION.
[FILE SECTION.
{ FD    nume-de-fişier
                                 {V}
        [RECORDING MODE IS       {F}]
                                 {U}
                                                  {RECORDS   }
        [BLOCK CONTAINS întreg-1 [TO întreg-2]    {CHARACTERS}]
        [RECORD CONTAINS întreg-3 [TO întreg-4] CHARACTERS]
                                 {STANDARD                              }
        LABEL {RECORDS ARE}      {OMITTED                               }
              {RECORD IS  }      {nume-de-dată-1[nume-dată-2] ...       }
        [DATA {RECORDS ARE}  nume-articol-1 [nume-articol-2] ...].
              {RECORD IS  }
  Ø1    nume-articol-1.
  .
  .
  .
  Ø1    nume-articol-2.
  .
  .
  .
  SD    nume-fişier-3
```

```
            DATA {RECORDS ARE} nume-articol-3 [nume-articol-4] ....
                 {RECORD IS   }

   01       nume-articol-3.
   .
   .
   .
   01       nume-articol-4.
   .
   .
   .
   FD       nume-fişier-la-imprimantă

            [RECORD CONTAINS întreg-7 TO întreg-8] CHARACTERS]

            [{REPORT IS  }
             {REPORTS ARE} tabel-1[tabel-2] ...].
```

8

```
[ WORKING-STORAGE SECTION.
  [ 77   nume-dată                      clauze-de-descriere-a-datelor.] ...
    [88  nume-condiţie                  clauza VALUE.] ...
  [ 01   nume-articol                   clauze-de-descriere-a-datelor.
    [{02}
     {49} nume-dată                     clauze-de-descriere-a-datelor.] ...
    [88  nume-condiţie    clauza-VALUE.] ...                              ] ]

[ LINKAGE SECTION.
  [aceeaşi structură ca la WORKING-STORAGE SECTION] ]
```

Rubricile de descriere a datelor pot avea unul din formatele:

Format-1

```
număr-de-nivel {nume-de-dată-1}
               {FILLER        }

[REDEFINES nume-de-dată-2]

[{PICTURE}
 {PIC    } IS lanţ-de-caractere]
```

```
[           { DISPLAY                                ]
[           { {COMPUTATIONAL }                       ]
[           { {COMP          }                       ]
[           { {COMPUTATIONAL-1}                      ]
[ USAGE IS  { {COMP-1         }                      ]
[           { {COMPUTATIONAL-2}                      ]
[           { {COMP-2         }                      ]
[           { {COMPUTATIONAL-3}                      ]
[           { {COMP-3         }                      ]
[           { INDEX                                  ]

[{SYNC         } {LEFT }]
[{SYNCHRONIZED } {RIGHT}]

[{JUST     } RIGHT ]
[{JUSTIFIED}       ]

[BLANK WHEN ZERO]

[{VALUES ARE} literal-1 [{THROUGH} literal-2]]
[{VALUE IS  }           [{THRU   }          ]]

 [OCCURS {întreg-1 TIMES                                            }
 [       {întreg-1 TO întreg-2 TIMES DEPENDING ON nume-dată-3       }

 [INDEXED BY nume-index-1 [nume-index-2] ... ]].
```

Format-2

```
88        nume-de-condiție  {VALUE IS  } literal-1 [THRU literal-2].
                            {VALUES ARE}
```

Format-3

```
66        nume-de-dată-1

   RENAMES nume-de-dată-2 [{THRU   } nume-de-dată-3].
                          [{THROUGH}               ]
```

8 12

[REPORT SECTION,

RD nume-de-situație

[WITH CODE identificator-1]

```
[ CONTROL IS    ] { FINAL                                          } ]
[ CONTROLS ARE  ] { nume-dată-1 [nume-dată-2] ...                  } ]
                  { FINAL nume-dată-3 [nume-dată-4] ...            } ]

[ PAGE { LIMIT IS   } întreg-1 { LINE  } [HEADING întreg-2]
[      { LIMITS ARE }          { LINES }

[FIRST DETAIL întreg-3 [LAST DETAIL întreg-4] [FOOTING întreg-5]

Număr de nivel [ { nume-grup-1 } ]
               [ { nume-rind-1 } ]

[           { { REPORT HEADING  }                                  } ]
[           { { RH              }                                  } ]
[           { { PAGE HEADING    }                                  } ]
[           { { PH              }                                  } ]
[           { { CONTROL HEADING }   { identificator-2 }            } ]
[           { { CH              }   { FINAL           }            } ]
[ TYPE IS   { { DETAIL }                                           } ]
[           { { DE     }                                           } ]
[           { { CONTROL FOOTING }   { identificator-3 }            } ]
[           { { CF              }   { FINAL           }            } ]
[           { { PAGE FOOTING    }                                  } ]
[           { { PF              }                                  } ]
[           { { REPORT  FOOTING }                                  } ]
[           { { RF              }                                  } ]

[ LINE  NUMBER  IS { întreg-6 [ON NEXT PAGE] } ]
[                  { PLUS întreg-7           } ]
[                  { ON NEXT PAGE            } ]

[ NEXT GROUP IS { întreg-8      } ]
[               { PLUS întreg-9 } ]
[               { ON NEXT PAGE  } ]

[BLANK WHEN ZERO]

[COLUMN NUMBER IS întreg-10]
```

[GROUP INDICATE]

[{ JUSTIFIED / JUST } RIGHT]

8 12

[{ PICTURE / PIC }] IS lanț – de – caractere [USAGE IS DISPLAY]

[[RESET ON { nume – dată – 5 / FINAL }]]

[{ SOURCE IS { identificator – 4 / TALLY }

SUM { identificator – 5 / TALLY } [{ identificator – 6 / TALLY }]

[UPON nume – de – dată – 6 [nume – de – dată – 7] ...

[VALUE IS literal – 1] }] .

PROCEDURE DIVISION [USING nume – dată – 1 [nume – dată – 2] ...].

| Secțiunea declarativelor |

[DECLARATIVES.

{nume – secțiune SECTION.

{ USE {AFTER} STANDARD [{ BEGINNING / ENDING }] [{ REEL / FILE / UNIT }]

[redacted]

LABEL PROCEDURE ON { nume – fișier – 1 [nume – fișier – 2] ... / INPUT / OUTPUT / { INPUT – OUTPUT / I – O } }

USE AFTER STANDARD ERROR PROCEDURE

ON { nume – fișier – 1 [nume – fișier – 2] ... / INPUT / OUTPUT / { INPUT – OUTPUT / I – O } } }

```
    | USE BEFORE REPORTING nume-de-dată
    |            {[nume-secţiune SECTION.]
    |            {nume-paragraf. {frază} ... } ... } ...

END RECLARATIVE.]
```

| INSTRUCŢIUNI aritmetice |

Instrucţiunea de adunare (ADD)

Format — 1

```
ADD {identificator-1} [[{identificator-2}]] ... TO
    {literal-1      } [[{literal-2      }]]
    identificator-n  [identificator-m] ...
    [ROUNDED]
    [ON SIZE ERROR instrucţiune-imperativă ....]
```

Format — 2

```
ADD {identificator-1} {identificator-2} ...
    {literal-1      } {literal-2      }
    GIVING identificator-m
    [ROUNDED]
    [ON SIZE ERROR instrucţiune-imperativă ....]
```

Format — 3

```
ADD {CORRESPONDING} identificator-1 TO identificator-2
    {CORR         }
    [ROUNDED]
    [ON SIZE ERROR instrucţiune-imperativă ....]
```

Instrucţiunea de scădere (SUBTRACT)

Format — 1

```
SUBTRACT {literal-1      } {literal-2      } ...
         {identificator-1} {identificator-2}
         FROM identificator-m
         [ROUNDED]
         [ON SIZE ERROR instrucţiune imperativă ....]
```

Format — 2

```
SUBTRACT {literal-1      } {literal-2      } ...
         {identificator-1} {identificator-2}
```

FROM {literal–m | identificator–m} **GIVING** identificator.

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Format 3

SUBTRACT {[**CORRESPONDING** | [**CORR**} identificator–1

FROM identificator–2

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Instrucţiunea de înmulţire (*MULTIPLY*)

Format–1

MULTIPLY {identificator–1 | literal–1} **BY** identificator–2

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Format–2

MULTIPLY {identificator–1 | literal–1} **BY** {identificator–2 | literal–2}

GIVING identificator–3

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Instrucţiunea de împărţire (*DIVIDE*)

Format–1

DIVIDE {identificator–1 | literal–1} **INTO** identificator–2

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Format–2

DIVIDE {identificator–1 | literal–1} **INTO** {identificator–2 | literal–2}

GIVING identificator–3

[**ROUNDED**]

[ON **SIZE ERROR** instrucţiune–imperativă . . . .]

Format—3

DIVIDE {identificator—1 / literal—1} **BY** {identificator—2 / literal—2}

GIVING identificator—3
[**ROUNDED**]
[**REMAINDER** identificator—4]
[ON **SIZE ERROR** instrucţiune—imperativă . . . .]

Instrucţiune pentru calcule complexe (COMPUTE)

COMPUTE identificator—1 [**ROUNDED**] = {literal / identificator—2 / expresie—aritm}

[ON **SIZE ERROR** instrucţiune—imperativă . . . .]

Expresie aritmetică

[OS] [(. . . {id—el—nr / literal—nr} {{** / / / * / + / —} [(. . .] {id—el—nr / literal—nr}} . . . [)] . . .]

ORDINEA DE EFECTUARE A OPERAŢIILOR ARITMETICE

1	OS	operator—semn (minus)
2	**	exponent — ridicare la putere
3	*	înmulţire
	/	împărţire
4	+ —	adunare scădere
.	SD	ordine stinga—dreapta în cazul operatorilor de acelaşi nivel

Instrucţiuni de lucru cu fişiere

Instrucţiunea de deschidere a fişierelor (OPEN)

OPEN [**INPUT** {nume—de—fişier [**REVERSED** / WITH **ND REWIND**]} . . .]

[**OUTPUT** {nume—de—fişier [WITH **NO REWIND**]} . . .]

```
[ { INPUT—OUTPUT }  {nume—de—fişier...} ]
  { I—O          }
```

Instrucţiunea de închidere a fişierelor (CLOSE)

```
CLOSE nume—de—fişier—1 [ REEL ] [ WITH { NO REWIND } ]
                       [ UNIT ] [      { LOCK      } ]

   [ nume—de—fişier—2 [ REEL ] WITH { NO REWIND } ] ]
   [                  [ UNIT ]      { LOCK      } ] ]
```

Instrucţiunea de citire a unui articol dintr-un fişier (READ)

```
READ  nume—de—fişier RECORD [INTO nume—dată]
        { AT END      }
        { INVALID KEY } instrucţiune—imperativă....
```

File control	ACCESS	S	S	R	R
	ORGANIZATION	S	IS	D	IS
	ACTUAL KEY	—	—	DA	—
	RECORD KEY	—	—	—	DA
	SYMBOLIC KEY	—	—	DA	DA
OPEN	CONSULTARE—CITIRE	I	I	I	I
	ACTUALIZARE DISC	I—O	I—O	I—O	I—O
READ	INTO	DA	DA	DA	DA
	AT END	DA	DA	—	—
	INVALID KEY	—	—	DA	DA

Instrucţiunea de scriere a unui articol într-un fişier (WRITE)

```
WRITE  nume—de—articol [FROM nume—de—dată]
  [ { INVALID KEY instrucţiune—imperativă ....                       } ]
  [ { { BEFORE } ADVANCING { identificator—LINES                    } } ]
  [ { { AFTER  }           { întreg LINES                           } } ]
  [ {                      { nume—mnemonic                          } } ]
```

		CREARE DE FIŞIER COMPLETARE FIŞIER							
CONTROL FILE	ACCESS	S	S	R	R	S	S	R	R
	ORGANIZATION	S	IS	D	IS	S	IS	D	IS
	ACTUAL KEY	—	—	DA	—	—	—	DA	—
	RECORD KEY	—	DA	—	—	—	—	—	DA
	SYMBOLIC KEY	—	—	DA	—	—	—	DA	DA
	ENUNT OPEN	I	I	I	—	—	—	I—O	I—O
WRITE	FROM	DA	DA	DA	—	—	—	DA	DA
	BEFORE AFTER	*DA	—	—	—	—	—	—	—
	INVALID KEY	—	DA	DA	—	—	—	DA	DA

* numai pentru imprimare

Instrucţiune pentru rescrierea unui articol într-un fişier (REWRITE)

REWRITE nume—articol[**FROM** nume—de—dată]

[**INVALID** KEY instrucţiune—imperativă . . . .]

Instrucţiune pentru căutarea articolului sau a amplasamentului său într-un fişier cu organizare selectivă (SEEK).

SEEK nume—fişier RECORD

Instrucţiune pentru anularea unui articol dintr-un fişier (DELETE)

DELETE nume—articol

Instrucţiuni utilizate în operaţii de sortare a fişierelor

Instrucţiunea pentru efectuarea sortării (SORT)

SORT nume—de—fişier—1 ON { **DESCENDING** | **ASCENDING** } KEY

identificator—1

[identificator—2]. . .

[ON { **ASCENDING** | **DESCENDING** } KEY identificator—3

[identificator—4] . . .] . . .

{ **INPUT PROCEDURE** IS nume—secţiune—1 [**THRU** nume—secţiune—2] | **USING** nume—fişier—2 }

{ **OUTPUT PROCEDURE** IS nume—secţiune—3 [**THRU** nume—secţiune—4] | **GIVING** nume—fişier—3 }

Instrucţiunea pentru încărcarea unui articol în fişierul intermediar de sortare

RELEASE nume—articol [**FROM** identificator]

Instrucţiunea care redă disponibil un articol al fişierului intermediar de sortare (RETURN)

RETURN nume—de—fişier [**INTO** identificator] AT **END**
instrucţiune—imperativă . . . .

Instrucţiuni de intrare-ieşire în condiţiile utilizării unui volum mic de date

Instrucţiunea de introducere a datelor în memoria calculatorului (ACCEPT)

ACCEPT nume—dată { [**FROM** { **CONSOLE** | nume—mnemonice }] | **FROM** { **DATE** | **DAY** | **TIME** } }

Instrucţiunea de extragere a datelor din memorie (*DISPLAY*)

DISPLAY { literal – 1 | identificator – 1 } [{ literal – 2 | identificator – 2 }] ...

[**UPON** { **CONSOLE** | **SYSPUNCH** | nume – mnemonice }]

Instrucţiuni de transfer

Instrucţiunea de transfer (*MOVE*)

Format – 1

MOVE { identificator – 1 | literal | constantă – figurativă } **TO** identificator – 2 ...

Format – 2

MOVE { **CORRESPONDING** | **CORR** } identificator – 1 **TO** identificator – 2

Tipul datei			Tipul transferului			
			a	b	c	d
Data compusă		a	A	A	A*	A*
Dată elementară	NENUMERICĂ	b	A	A	A*	A*
	NUMERICA	c	A	Ac	Nc	Ec
	EDITARE	d	A	A	interzis	

A — transfer ALFANUMERIC
A* — transfer alfanumeric cu rezultat imprevizibil dacă zona de emisie nu conţine numere cu semn
Ac — Transfer ALFANUMERIC cu conversie de caractere
Nc — transfer NUMERIC cu conversie
Ec — transfer de EDITARE cu inserţii şi substituiri.

Instrucţiunea de examinare a cîmpurilor (*EXAMINE*)

Format – 1

EXAMINE identificator **TALLING** { **UNTIL** *FIRST* | **ALL** | **LEADING** } { literal – 1 | identif – 1 }

[**REPLACING BY** { literal – 2 | identificator – 2 }]

Format—2

EXAMINE identificator **REPLACING** { **ALL** | **LEADING** | **UNTIL FIRST** }

{ literal—1 | identificator—1 } **BY** { literal—2 | identificator—2 }

Instrucţiuni de control—prelucrare

Instrucţiunea de salt (GO TO)

GO TO [nume—procedura] [... **DEPENDING** ON identif]

Instrucţiunea de modificare (ALTER)

ALTER {nume—paragraf **TO**[**PROCEED TO**]nume—parag] ...

Instrucţiunea de identificare a unui punct de ieşire (program extern)

EXIT [PROGRAM].

Instrucţiunea de marcare a sfîrşitului unei lucrări (STOP)

STOP { literal | **RUN** }

Instrucţiunea de testare a unei condiţii (IF)

IF { cond—relaţie | cond—semn | nume—clasă | nume—condiţie | cond—compusă } { enunţ—cond—ADEV | **NEXT SENTENCE** } { **ELSE** enunţ—cond—FALSA | **ELSE NEXT SENTENCE** }

Condiţie de relaţie

{ identificator | literal | expr—aritm. } IS[**NOT**] { **GREATER** THAN | **LESS** THAN | **EQUAL TO** } { > | < | = } { identificator | literal | expr—aritm. }

COMPATIBILITĂŢI

Tipul testului	Situaţia identificatorilor		
	Tipul operaţiilor		USAGE
ALGEBRIC	nr. elem.	nr. elem.	oricare
NENUMERIC	nr. elem.	alte	DISPLAY
	alte	alte	

Condiţie de semn

{ identificator–elem–numeric / expresie–aritmetică } IS [NOT] { **POSITIVE** / **NEGATIVE** / **ZERO** }

Condiţie de clasă

{ identif–nr–DISPLAY / identif–nenumeric } IS [NOT] { **NUMERIC** / **ALPHABETIC** }

Condiţie compusă

[**NOT**] condiţie–A { **AND** / **OR** } [**NOT**] condiţie–B

Instrucţiuni de apelare a unei secvenţe

Instrucţiunea de apelare a unei secvenţe (*PERFORM*)

Format–1

PERFORM nume–secvenţă

Format–2

PERFORM nume–secvenţă { identificator–1 / intreg–1 } **TIMES**

Format–3

PERFORM nume–secvenţă **UNTIL** condiţie–1

Format–4

PERFORM nume–secvenţă

VARYING contor–1 **FROM** valoare–iniţială–1 **BY** pas–1
UNTIL condiţie–1
[**AFTER** contor–2 **FROM** valoare–iniţială–2 **BY** pas–2
UNTIL condiţie–2
[**AFTER** contor–3 **FROM** valoare–iniţială–3 **BY** pas–3
UNTIL condiţie–3]].

Format–5

PERFORM nume–secvenţă–1 **THRU** nume–secvenţă–n

Instrucţiunea de apel a unui subprogram (CALL)

CALL nume–subprogram [**USING** identificator–1 [identificator–2] . . .].

Instrucţiuni de editare (utilizînd editorul COBOL)

Instrucţiunea de iniţializare a contorilor asociaţi unei situaţii la imprimantă

INITIATE nume—situaţie—1 [nume—situaţie—2] ...

Instrucţiunea de generare a unei situaţii

GENERATE identificator

Instrucţiunea de completare a imprimării unei situaţii

TERMINATE nume—situaţie—1 [nume—situaţie—2] ...

Instrucţiuni de exploatare a tablourilor

Instrucţiunea pentru căutarea elementelor unui tablou (SEARCH)

Format—1

```
                                  { nume—index    }
SEARCH nume—dată—1 [VARYING       {               }
                                  { nume—dată—2   }
        [AT END instrucţiune—1]
                           { instrucţiune—2  }
        [WHEN condiţie—1   {                 }
                           { NEXT SENTENCE   }
                           { instrucţiune—3  }
        [WHEN condiţie—2   {                 } ]...
                           { NEXT SENTENCE   }
```

Format—2

```
SEARCH ALL nume—dată—1 [AT END instr—1 ...]
                        { instrucţiune—2  }
        [WHEN condiţie  {                 } ]
                        { NEXT SENTENCE   }
```

Instrucţiunea pentru iniţializarea şi avansarea numelor de index (la un tablou)

Format—1

```
     { nume—index—1 [nume—index—2]...               }
SET  {                                              } TO
     { identificator—1 [identificator—2] ...        }

     { nume—idex—3       }
     { identificator—3   }
     { literal—1         }
```

Format—2

```
                                         { UP BY   }
SET nume—index—4 [nume—index—5] ...      {         }
                                         { DOWN BY }

     { identificator—4 }
     {                 }
     { literal—2       }
```

Alte instrucţiuni COBOL

Facilităţi de punere la punct a programelor. Instrucţiunea (PRINT).

PRINT [nume−dată / literal] [,nume−de−dată / ,literal] ...

forma de ieşire: nume−paragraf {: nume−dată−valoare: valoare−literal}

Instrucţiunea de introducere a comentariilor (NOTE)

NOTE comentarii.

Instrucţiunea pentru extragerea dintr-o bibliotecă sursă a unui program COBOL

COPY { 'DV: periferic / 'DVT: tip−periferic,VS: indicator−volum} },

FN: 'nume−fişier sau/numele−cărţii', VN: nr−versiune,

{ GN: nr−generaţie / CD: data−creare } [,UN: nr−punere−la−zi],

[FS: identificator−volum],

[ON: 'nume−proprietar'],

[VP: 'parolă−volum'],

[FP: 'parolă−fişier'],

[*REPLACING* cuvînt−1 *BY* cuvînt−2]

[cuvînt−3 *BY* cuvînt−4] ...].

LISTA BILINGVĂ A CUVINTELOR REZERVATE COBOL

ACCEPT — primeşte
ACCESS— acces
ACTUAL — actual
ADD — adună
ADDRESS — adresă
ADVANCING — avansînd
AFTER — după
ALL — tot
ALPHABETIC — alfabetic
ALTER — schimbă—modifică
ALTERNATE — alternativ
AN — un, o
AND — şi
APPLY — solicită, aplică
ARE — sînt
AREA — cîmp, arie, zonă
AREAS — cîmpuri, arii, zone
ASCENDING — crescător
ASSIGN — repartizează, atribuie
AT — pînă la, la, spre
AUTHOR — autor
BEFORE — înainte de
BEGINNING — începînd
BLANK — blanc, spaţiu
BLOCK — bloc
BY — prin, cu
CALL — cheamă
cancel — întrerupere, anulare
CF
CH
CHANNEL — canal
CHARACTERS — caractere
CLOCK — UNITS — dispozitiv măsurare timp
CLOSE — închide
COBOL — COBOL
CODE — cod, cifru
COLUMN — coloană
COMMA — virgulă
COMP — calcul în binar

COMP—1 — calc.vg.mob.simp.
COMP—2 — calc.vg.mob.dbl.
COMP—3 — calcul în zecimal împachetat
COMPUTATIONAL — COMP
COMPUTATIONAL—1 — COMP—1
COMPUTATIONAL—2 — COMP—2
COMPUTATIONAL—3 — COMP—3
COMPUTE — calculează
CONFIGURATION — configuraţie
CONSOLE — maşină de scris
CONSTANT — constant
CONTAINS — conţine
CONTROL — control
CONTROLS — verificări
COPY — copiază
CORR
CORRESPONDING — corespunzător
CURRENCY — monedă, valută
DATA — dată (informaţie)
DATE—COMPILED — data (zi, lună,) compilării
DATE—WRITTEN — data (zi, luna) scrierii
DE — vezi DETAIL
DECIMAL—POINT — virgulă zecimala notată prin punct
DECLARATIVES — declaraţii, text.
DELETE — şterge
DEPENDING — depinzînd de
DESCENDING — descrescător
DETAIL — detaliu
DIRECT — direct
DISPLAY — afişează
DIVIDE — imparte
DIVISION — diviziune, parte
DOWN — în jos vertical
ELSE — altfel
END — sfîrşit
END—OF—PAGE — sfîrşit de pagină
ENDING — terminaţie
ENTER — intrare, introducere
ENVIRONMENT — mediu, cadru
EOP — sfîrşit de pagină
EQUAL — egal
EQUALS — egali
ERROR — eroare
EVERY — fiecare, toţi
EXAMINE — examinează
EXCEEDS — depăşeşte
EXIT — ieşi
FD — indicator-nivel pentru descrierea fişierelor
FILE — fişier
FILE—CONTROL — control — fişiere
FILE—LIMIT — limită de fişier
FILE—LIMITS — limite de fişiere
FILLER — denumire pentru arii inactive, neadresabile, nedefinte
FINAL — sfîrşit
FIRST —
FOOTING — pozitive, nivel, bază
FOR — pentru
FROM — de la, din
GENERATE — generează
GIVING — transmiţind
GO — mergi, du-te
GREATER — mai mare
GROUP — grupează
HASHED — fărămiţat
HEADING — titlu, cap
HIGH—VALUE — valoare maximă
HIGH—VALUES — valori maxime
HOLD — magazie
IDENTIFICATION — identificare
IF — dacă
IN — în, la, cu
INDEX — indice
INDICATE — indică
INITIATE — iniţiază
INPUT — intrare
INPUT—OUTPUT — intrare—ieşire
INSTALLATION — echipament, utilaj
INTO — prin, în
INVALID — invalid
I—O — intrare—ieşire
I—O — CONTROL — controlul intrare—ieşire
IS — este
JUST — aliniat
JUSTIFIED — aliniat
KEY — cheie, cod
KEYS — chei, coduri
LABEL — etichetă
LABELS — etichete
LAST — ultim
LEADING — conducător
LESS — mai mic
LEFT — stînga
LIBRARY — bibliotecă
LIMIT — limită
LIMITS — limite
LINAGE — liniatură
LINAGE—COUNTER — control de liniatură
LINE — linie
LINE—COUNTER — contor de linii
LINES — linii
LINKAGE — asamblare
LOCK — blochează
LOW—VALUE — valoare minimă
LOW—VALUES — valori minime
LOWER—BOUND — granița de jos
LOWER—BOUNDS — graniţele de jos
MEMORY
MODE — mod, format
MODULES — module
MOVE — transfera
MULTIPLE — multiplu
MULTIPLY — înmulţeşte
NEGATIVE — negativ
NEXT — următor
NO — nu

NOT — nu
NOTE — note
NUMBER — număr
NUMERIC — numeric
OBJECT—COMPUTER — calculator folosit la execuţie
OCCURS — repetări, apariţii
OF — de, de la, din
OFF — din afară
OH
OMITTED — omite
ON — pe, la
OPEN — deschide
OPTIONAL — facultativ
OR — sau
ORGANIZATION — organizare
OUTPUT — ieşire
OV
OVERFLOW — depăşire
PAGE — pagină
PAGE—COUNTER — contor pag.
PERFORM — execută grup instrucţiuni cu reîntoarcere
PF
PH
PIC
PICTURE — imagine, descriere
PLUS — plus
POSITION — poziţie
POSITIVE — pozitiv
PREPARED — preparat
PRINT — imprimă
PRIORITY — priotitate
PROCEDURE — prelucrare, procedură
PROCEED — continuă, reia
PROCESS — prelucrare, operaţie
PROCESSING — prelucrînd
PROGRAM — program
PROGRAM—ID — titlu—paragraf identificare program
QUOTE — ghilimea, apostrof
QUOTES — ghilimele
RANDOM — aleator
RANGE — rînd
RD — indicator nivel descrierea listării
READ — citeşte
RECORD — articol, înregistrare—logică
RECORDING — înregistrare, format articol
RECORDS — articole
REDEFINES — redefineşte
REEL — bobinează
RELEASE — eliberează
REMAINDER — rest, rămăşiţă
REMARKS — comentarii
RENAMES — redenumeşte
REPLACING — înlocuind
REPORT — report
REPORTING — reportînd (listar)
REPORTS — reporturi (listare)
RERUN — reia lucrul
RESERVE — rezervă
RESET — repune
RETURN — reîntoarce
REVERSED — întors
REWIND — rebobinează
REWRITE — rescrie
RF
RH
RIGHT — dreapta
ROUNDED — rotunjit
RUN — lucru execuţie
SA
SAME — acelaşi, aceeaşi
SD — indicator nivel SORT
SEARCH — cercetare
SECTION — secţiune, capitol
SECURITY — siguranţă
SEEK — caută
SEGMENT—LIMIT — limită—segment
SELECT — alege, selectează
SELECTED — ales, selectat
SENTENCE — propoziţie, frază
SEQUENTIAL — secvenţial
SET — grup
SIGN — semn, simbol
SIZE — mărime, dimensiune
SORT — sortează, triază
SOURCE — sursă, izvor
SOURCE—COMPUTER compilare
SPACE — spaţiu, blanc
SPACES — spaţii, blancuri
SPECIAL—NAMES — nume—
STANDARD — etalon, standard
STATUS — stare, situaţie
STOP — stop, opreşte
SUBTRACT — scade
SUM — sumă total
SUPERVISOR — supervizor
SUSPEND — suspendă
SYMBOLIC — simbolic
SYNC — aliniere în memorie
SYNCHRONIZED — idem
SYSIN — simb. echip—intr.
SYSOUT — simb. echip—ieşire
SYSPUNCH — simb. echip—perf.
SYSRERUN — echip—reluare
TALLY — registru de evidenţă
TALLING — evidenţiind
TAPE — bandă
TERMINATE — încheie, termină
THAN — decît
THEN — atunci, apoi, deci
THROUGH — prin
THRU — la, prin, pînă
TIMES — perioade, număr de
TO — la, spre, pînă la
TYPE — categorie
UNIT — unitate de prelucrare.

UNTIL — pînă la, pînă
UP — în sus, ascendent
UPON — pe
UPPER—BOUND — granița de sus
UPPER—BOUNDS — granița de sus
USAGE — întrebuințare, folosire
USE — folosește
USING — folosind
VALUE — valoare
VALUES — valori
VARYING — variind
WHEN — cînd, deși, după ce, pe, cînd
WITH — cu, și la, față de
WORDS — cuvinte
WORKING—STORAGE — memoria de lucru
WRITE — scrie
ZERO — zero
ZEROES — zerouri
ZEROS — zerouri

CODIFICAREA EBCDIC A SETULUI DE CARACTERE COBOL

Car.	Reprezentare binară în memorie	Val. zec.	Val. hex.	Cod. car
VAL MIN	0000 0000	000	00	
b̸	0100 0000	064	40	
.	0100 1011	075	48	12—3—8
<	0100 1100	076	4C	12—4—8
(	0100 1101	077	4D	12—5—8
+	0100 1110	078	4E	12—6—8
	0100 1110	078	4E	12—6—8
$	0101 1011	091	5B	11—3—8
=	0101 1100	092	5C	11—4—8
)	0101 1101	093	5D	11—5—8
;	0101 1110	094	5E	11—6—8
-	0110 0000	096	60	11
/	0110 0001	097	61	0—1
,	0110 1011	107	6B	0—3—8
>	0110 1110	110	6E	0—6—8
'	0111 1101	125	7D	5—8
=	0111 1110	126	7E	6—8
A	1100 0001	193	C1	12—1
B	1100 0010	194	C2	12—2
C	1100 0011	195	C3	12—3
D	1100 0100	196	C4	12—4
E	1100 0101	197	C5	12—5
F	1100 0110	198	C6	12—6
G	1100 0111	199	C7	12—7
H	1100 1000	200	C8	12—8
I	1100 1001	201	C9	12—9

Car.	Reprezentare binară în memorie	Val. zec.	Val. hex.	Cod. cart.
J	1101 0001	209	D1	11—1
K	1101 0010	210	D2	11—2
L	1101 0011	211	D3	11—3
M	1101 0100	212	D4	11—4
N	1101 0101	213	D5	11—5
O	1101 0110	214	D6	11—6
P	1101 0111	215	D7	11—7
Q	1101 1000	216	D8	11—8
R	1101 1001	217	D9	11—9
S	1110 0010	226	E2	0—2
T	1110 0011	227	E3	0—3
U	1110 0100	228	E4	0—4
V	1110 0101	229	E5	0—5
W	1110 0110	230	E6	0—6
X	1110 0111	231	E7	0—7
Y	1110 1000	232	E8	0—8
Z	1110 1001	233	E9	0—9
0	1111 0000	240	F0	0
1	1111 0001	241	F1	1
2	1111 0010	242	F2	2
3	1111 0011	243	F3	3
4	1111 0100	244	F4	4
5	1111 0101	245	F5	5
6	1111 0110	246	F6	6
7	1111 0111	247	F7	7
8	1111 1000	248	F8	8
9	1111 1001	249	F9	9
VAL MAX	1111 1111	255	FF	

ANEXA 2. Lista cu mesaje erori

LISTA ERORILOR SEMNALATE ÎN TRATAREA FIȘIERELOR

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAȚIA MESAJULUI
00	SEG.INCONNU	Apel greșit al unui segment SGF din zona tranzitorie. Eroare de programare sau memorie distrusă.
01	OUV. F. ACTIF	Încercare de deschidere a unui fișier deja deschis. Eroare de programare.
02	IDEX ERRONE	Idex neidentificat. Eroare de programare sau memorie distrusă.
03	OUV. MF. ACTI	Deschidere simultană a două fișiere dintr-o structură multifișier. Eroare de programare.
04	DVT ABSENT	Tipul aparatului nu a fost declarat. Eroare de programare.
05	PRM ABSENT	Modul de tratare nu a fost precizat. Eroare de programare.
06	UPD INTERD	Modul de tratare INPUT—OUTPUT este folosit pentru bandă, ceea ce este interzis. Eroare de programare.
07	DVT : TDF ≠ ASS	Perifericul a fost declarat de două ori cu tip diferit. Eroare de programare.
08	VOL1 EN NST	Tentativă de scriere nestandard pe o bandă organizată standard.
09	POSITION MT	Poziționarea greșită pentru scriere în multifișier. Eroare de programare.
0A 0B	MT. PROTEGE	Tentativă de scriere pe un volum protejat pentru scriere. Eroare de programare.
10	MFA INTERD	Declarație de zonă partajată pe bandă magnetică. Eroare de programare.
11	EOF1 ABSENT	Structura benzii nu corespunde fișierului organizat standard sau poziționarea, pentru citirea inversă sau pentru scrierea în continuare, este greșită.
12	VOL1 ABSENT	Tentativă de scriere sau citire standard pe un volum nestandard
13	ON INCONNU	Parametrul ON are valoarea de pe suport diferită de cea declarată pe cartela LABEL.
14	FM MAL PLACE	Structura benzii nu corespunde structurii standard a fișierului.
15	FS INCONNU	Tentativă de scriere într-un fișier la care ansamblul de volume nu a fost asociat.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAȚIA MESAJULUI
16	F. NON PERIME	Încercare de anulare a un fișieir neperimat.
17	HDR1 ABSENT	Structura benzii nu corespunde structurii standard a fișierului.
18	MFT : ER.DATE	Data de expirare a unui fișier din structura multifișier este mai mare decît a celui precedent.
19	COND. TDF INCONNU	Codul TDF diferit de SURI. Eroare de programare sau memorie distrusă.
20	PRM ERRONE	Modul de tratare a fost modificat sau distrus în timpul tratării.
21	FN INCONNU	Fișierul nu este prezent în volumul respectiv.
22	GEN INCONNU	Pe suport (bandă) nu este înregistrat numărul de generație al fișierului.
23	FV ERRONE	Numărul de ordine al primului volum nu corespunde cu cartela LABEL.
24	CD INCONNU	Data sau generația nu sînt prezente pe suport.
25	VN INCONNU	Numărul versiunii nu este prezent pe suport.
26	LAB2 ABSENT	Eticheta HDR2 sau EOF2 sau EOV2 nu este prezentă pe banda magnetică. Structura benzii nu este cea corespunzătoare sau banda este deteriorată.
27	SEQ.LPS ERR	Eroare în procedurile utilizator destinate tratării etichetelor.
28	DVF ERRONE	Parametrii argumentului DVF sînt eronați. Eroare de program sau memorie alterată.
29	F.NON TROUVE	Fișier neidentificat pe suport. Cartelele LABEL și ALLOC nu sînt compatibile sau volumul este greșit.
30	ZP IMPOSSIB	Tentativă de a crea un număr mai mare de zone partajate decît cel admis.
31	OFO : END/EXT	Scrierea articolului următor nu poate fi făcută în zona de extensie.
32	DVT ERRONE	Parametrul care reprezintă tipul perifericului a fost modificat în timpul tratării fișierului.
33	BFS ERRONE	BFS-ul declarat la exploatare diferă de cel declarat la creare. Eroare de programare.
34	ZP REMPLIE	Zona partajată este insuficientă. Nu se mai poate continua crearea fișierului.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAŢIA MESAJULUI
35	F.DEJA CREE	Tentativă de creare a unui fişier care a fost creat. Eroare de programare.
36	ZP SATUREE	Se încearcă crearea unui fişier într-o zonă partajată complet ocupată.
37	XU IMPOSSIB	Zona de extensie nu poate fi creată din lipsă se suport.
38	REFUSAL.NEW	Parametrul de alocare NEW a fost utilizat pentru un volum care nu este complet liber.
39	F.NON ECRIT	Încercare de deschidere pentru exploatarea unui fişier care nu a fost creat.
40	ORG.F.ERR	Organizarea fişierului nu este secvenţială înlănţuită. Controlul s-a făcut la nivelul etichetei HDR2. Eroare de programare.
41	LEC ERRONE	Eroare de intrare-ieşire ireparabilă, constatată la citirea etichetelor sau a fişierului FILCOM etc.
42	HDR3 ABSENT	Fişierul nu este prezent în zona partajată. Eroare de programare.
43	HDR1 ABSENT	La o încercare de deschidere a unui fişier, altul decît primul, dintr-o structură multifişier nu a fost găsită eticheta HDR1. Banda a fost greşit poziţionată sau structura ei nu este cea corespunzătoare.
44	F.NON ACTIF	S-a încercat închiderea unui fişier care nu a fost deschis. Eroare de programare sau memorie alterată.
45	PRM.OFO.ACM	Incompatibilitate între modul de tratare, condiţiile de deschidere şi modul de acces la fişier.
46	CARTE ERR	Întîlnirea unei cartele cu punct interzisă în fişier.
47	LAB1. ABSENT	În timpul închiderii unui fişier tratat prin citirea înainte, etichetele EOF1 sau EOV1 nu au fost găsite. Structura suportului este necorespunzătoare.
48	NB BL ERR	Blocurile fişierului nu au fost citite.
49	GN INCONNU	Numărul de generare nu este prezent pe volum.
4A	UCO ERRONE	Indicatorul de frecvenţă (indicatorul de actualizare,nu este specificat prin argumentul MFF al cartelei ! FILE sau în descrierea fişierului.
50	EOF ABSENT	Adresa secvenţei de tratare a sfîrşitului fişierului nu este prezentă în TDF. Eroare de programare.
51	E/S ERRONE	Eroare de intrare/ieşire ireparabilă, constatată la o instrucţiune OPEN sau CLOSE.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAŢIA MESAJULUI
52	FIN ABS. VOL	Alocarea este abandonată deoarece lipseşte volumul următor.
53	IDEX.ERR	Idexul din TDF nu este compatibil cu perifericul cerut pentru un fişier al sistemului.
54	FIN DE ZONE	A fost detectată apropierea de sfîrşit de zonă în timpul creării fişierului.
55	RES.GF.DETR	Tentativă de scriere în exteriorul zonei FILCOM.
56	ACM ≠ SEQ	Fişierul de recepţie a informaţiilor de repriză nu este secvenţial.
57	MFS ≠ UND	Fişierul de recepţie a informaţiilor de repriză nu este secvenţial continuu.
58	MOD ≠ STD	Fişierul de recepţie a informaţiilor de repriză este de organizare nestandard (pe bandă).
59	PRM ≠ OUT	Modul de tratare a fişierului de recepţie a informaţiilor de repriză nu este cel de scriere.
60	DVT INTERD	Tipul aparatului nu corespunde unui fişier de recepţie pentru informaţiile de repriză.
61	TOV SATURE	Tabelul de ocupare a volumului este complet.
62	XU REMPLIE	Zona de extensie este insuficientă (a fost umplută).
63	DVN : TDV ≠ ASS	Numărul de periferice în TDF este mai mic decît cel rezultat din cartela ASSIGN.
64	CHKP INTERD	Este interzisă citirea pentru utilizarea informaţiilor de repriză.
65	AC.FP.ERR	Eroare detectată în timpul tratării procedurii de protecţie a fişierului prin cuvîntul de trecere.
66	AC.VP.ERR	Eroare detectată în timpul tratării procedurii de protecţie a volumului prin cuvîntul de trecere.
67	SZ ABSENT	Dimensiunea zonei de alocat nu este specificată.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAŢIA MESAJULUI
68	EXCES ALLOC	Volum inutil alocat.
69	TRT LPS ERR	În procedura utilizator de tratare a etichetelor se utilizează greşit macroinstrucţiunile corespunzătoare.
6A	SATURATION	Tabela destinată controlului unei aceleiaşi zone utilizată în multiacces a fost umplută.
6B	NL ERRONE	Perifericul ales pentru fişier nu este asociat partiţiei în care se execută programul ce tratează fişierul respectiv.
6C	ERR PROTEC	Eroare de protecţie a unei zone solicitată de o partiţie.
6D	CKP.FM.	S-a detectat o marcă de fişier în interiorul unor informaţii de repriză.
6E	CLOSE I IMP	Suprimarea etichetei prin DELETE este cerută deoarece fişierul este vid. Închiderea este imposibilă.
70	COD TAB ERR	Este greşită adresa tabelului de definiţie afişierului.
71	NO.APP.ABS	Perifericul nu este disponibil.
72	LAB ERRONE	Eticheta citită de pe disc a fost greşită.
73	OFO : END/VOL	Volumul nu conţine ultimul bloc al fişierului.
74	ASSIGN INC	Asocierea aparatului este incorectă înaintea lansării unei reluări.
75	BFC ERRONE	Este interzisă gestiunea programată a buferilor. Eroare de programare.
76	VOLS.INSUF.	Dimensiunea alocată este insuficientă pentru a conţine toate casetele fişierului selectiv sau lipseşte un volum pentru a realiza adaptarea.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAȚIA MESAJULUI
77	DIMENSION ALLOUEE INSUFFISANTE. MONTAGE VOL.SUPP.POUR ALLOC OUI ? NON	Suportul alocat este insuficient. Alocarea este posibilă prin montarea unui volum suplimentar pe unitatea periferică specifică în mesajul transmis.
78	RCF ERRONE	La exploatarea fișierului articolul are format diferit de cel avut la creare.
79	KYS ERRONE	Valoarea cheii utilizate la exploatarea fișierului nu este compatibilă cu dimensiunea cheii fixată la creare.
80	BFN = 1.CP 300	BFN nu poate fi egal cu 1.
81	DEB.NO.MONT	În timpul deschiderii a fost constatată o montare greșită a volumului și anume volumul cu partea de depășire a fișierului nu a fost montat.
82	WFLM INTERD	Utilizarea macroinstrucțiunii WFLM este interzisă.
83	PRM : UND INT	Modul de tratare ales este interzis pentru organizarea selectivă. Eroare de programare.
84	BFS INC.RCS	Mărimea articolului este incompatibilă cu dimensiunea zonei tampon. Eroare de programare.
85	IYY ERRONE	Numărul caracterelor suplimentare (hors texte) este insuficient. Eroare de programare.
86	ER.SEQ.VOL	Eroare de secvență la montarea volumelor.
87	BIS ERRONE	În exploatare, mărimea buferului este diferită de cea utilizată la crearea fișierului.
88	KYL ERRONE	Poziționarea cheii în interiorul articolului este incorectă. Eroare de programare.
89	EXT OUT DEB	Zona de extensie și zona de depășire sînt pe volume diferite.
8A	OPT.INTERD	Utilizarea macroinstrucțiunii FEOV este interzisă.
90	ER.MONT.VOL	Eroare în montarea volumului.

NUMĂRUL ERORII	TEXTUL MESAJULUI	EXPLICAȚIA MESAJULUI
91	BIN < 2	La crearea fișierelor secvențiale indexate, BIN, numărul zonelor tampon pentru index trebuie să satisfacă relația : 2<BIN⩽5. Această relație nu este satisfăcută. Eroare de programare.
92	DIM.CASE ER	Dimensiunea casetei este eronată
93	BIS < 256	Dimensiunea buferilor pentru index a fișierului de organizare selectivă este mai mică decît 256.
94	MFS ERRONE	În macroinstrucțiunea FEOV, opțiunile MFA sau MFT sînt interzise.
95	A.F.B.*3	În exploatarea fișierului sistemului *3 au fost detectate informații A.F.B.
96	SSF * CLOSE?	Se cere deschiderea unui subfișier * neînchis.
97	SSF * OPEN?	Se cere închiderea unui subfișier * nedeschis.
98	RCS ERRONE	Dimensiunea articolului este incorectă. Eroare de programare.
99	SGF.ME	Cerere greșită de închidere a fișierului de ieșire-monitor.
9A	TOE DETRUITE	În tratarea subfișierelor * 2 pe suporturi magnetice a fost distrus TOE
9B	F.M/ * IMT	În exploatarea fișierului de lucrări de pe bandă s-a detectat o marcă de fișier.
9C	FIN F/ * 1 RD	În exploatarea fișierului de lucrări de pe disc s-a detectat un sfîrșit de pagină.
9D	HDR1/ * 2	Adresa sector a etichetei HDR1 a fișierului de ieșire nu a fost comunicată de monitor.
9E	FIN Z./ * 2	Capacitatea afectată fișierului de ieșire pe disc a fost depășită.

LISTA PENTRU ALTE TIPURI DE ERORI.

1. Erori de sintaxă ale cartelelor de comandă

TEXTUL MESAJULUI	ACŢIUNEA SISTEMULUI	TIPUL CARTELEI
TYPE D'OPERATION NON IDENTIFIE	ABANDON DU TRAVAIL	Toate tipurile
ETIQUETTE ERRONEE	,,	
CARTES CONTROLE MAL ORDONEES	,,	
MOT CLE ERRONE	,,	Cartele relative la exploatarea şi la lansarea procesoarelor
ARGUMENT ERRONE	,,	
ABSENCE D'UN ARGUMENT OBLIGATOIRE	,,	
CHAINE DE CARACTERE ERRONEE	,,	
CARTE SUITE NON AUTORISEE	,,	
MOT CLE INCONU	,,	
ABSENCE DE SEPARATEUR APRES UN MOT CLE	,,	
ABSENCE DES DEUX POINT A LA SUITE D'UN ARGUMENT	,,	
ERREUR SUR LE PARAÎETRE DE L'UN DES ARGUMENTS	,,	
SYNTAXE DE L'ARGUMENT FS OU VS ERRONE	,,	Cartele relative la tratarea fişierelor
ABSENCE DU GUILLEMENT D'OUVERTURE DE L'ARGUMENT FN, UP, PP, OU, ON	,,	
ABSENCE DU GUILLEMENT DE FERMETURE DE L'ARGUMENT PN, VP, FP, OU, ON	,,	
ARGUMENT DU ABSENT	,,	
AFFECTATION D'APPAREIL IMPOSSIBLE	,,	
VALEUR DE L'ARGUMENT MD IMPOSIBLE	,,	
SYNTAXE ERRONEE EN FIN D'ARGUMENT OU EN FIN DE CARTE	,,	
PREMIERE LETTRE D'UN ARGUMENT AU DE LA COLONNE 68	,,	
DEPASSEMENT DE LA COLONNE 71 de LA CARTE	,,	
ASTERISQUE HORS COLONNE 72	,,	
OPTION DEMANDE DANS ALLOC AUTRE QUE NEW, ANY, SPC	—	
RANCONTRE DANS ARGUMENT DECIMAL DE CARACTERS NON DECIMAL	—	
VL COUPE ET ABSENCE DE SEPARATEUR DEMANDANT UNE CARTE SUITE	—	

2. Erori de încărcare a unui modul BT

TEXTUL MESAJULUI	ACȚIUNEA SISTEMULUI
MODULE OBJET ERRONE (INDICATIF DE TYPE D'ARTICLE)	ABANDON DU TRAVAIL
MODULE OBJECT BINAIRE ERRONE (CLE CHECKSUM)	,,
MODULE OBJET BINAIRE ERRONE (NUMERO DE SEQUENCE)	,,
MODULE OBJECT BINAIRE (NOM)	,,
EDITION DES LIENS IMPOSSIBLE	,,
CHARGEMENT IMPOSSIBLE	—

3. Erori de lansare a programului

TEXTUL MESAJULUI	EXPLICAȚIA MESAJULUI
PROGRAMME NON TROUVE DANS LA BIBLIOTHEQUE UTILIZATEUR STANDARD ERREUR DANS LANCEMENT DE PROGRAMME D'APRES POINT DE REPRISE	
IMPOSSM	Memorie insuficientă.
IMPOSSD	Periferice insuficiente.
IMPOSSD	Program inexistent în biblioteca standard.
IMPOSSF	Zona FILECOM este saturată.
IMPOSSL	Modulele SGF nu sînt incorporate.
IMPOSSC	Cartela relativă la tratarea fișierelor este invalidă.

4. Erori de execuţie a programului

FORMATUL MESAJULUI	ARGUMENTUL CCCC	EXPLICAŢIA MESAJULUI
ERREUR PROGRAMME 0000 CCCC	0101 0102 0104 0108 0201 0202 0204 0208 0400 0800	Depăşire binară Depăşire zecimală Depăşire flotantă (limita inferioară) Depăşirea flotantă (limita superioară) Instrucţiune interzisă (privilegiată) Adresă inexistentă Violarea memoriei Timpi de execuţie depăşiţi (la o instrucţiune) Instrucţiune inexistentă Eroare de sistem
	ARGUMENTUL YYXXXXX	
ERREUR E/S YYXXXXXX	01 a. CSV + 4 02 a.CB 03 a.CB 04 a.CB 05 a.CB 06 a.CB 07 a.CB 08 a.CSV + 4 09 a.CB 10 a.Cb 11. a.CB 12 a.CB 13 a.CB 14 a.CB 15 a.CB 17 a.CB	Adresa CB este invalidă Adresa PUE este invalidă Aparat periferic neafectat Adresa zonei tampon este eronată Eroare de programare E/S pe disc Zona disc este nealocată Numărul logic al perifericului este invalid Eroare în opţiunea RST Poziţionare fără întrerupere Cartela JOB citită de utilizator Lipsa zonei alocate Depăşirea numărului de linii Depăşirea zonei disc Numărător de octeţi invalid Urmează sfîrşitul de bandă Incident ireparabil de bandă

SISTEMUL DE 7 SERII ŞI COLECŢII ÎN AUTOMATICĂ-INFORMATICĂ ELECTRONICĂ-MANAGEMENT

I. SERIA „BIBLIOTECA DE AUTOMATICĂ-INFORMATICĂ-ELECTRONICĂ-MANAGEMENT

— Teme cuprinzătoare, reprezentative
— Formalism matematic cu expunere concisă, riguroasă, dar accesibilă
— Traduceri de mare notorietate
— Lucrări originale ale profesorilor, cercetătorilor, specialiştilor români de prestigiu
— Abordare multidisciplinară

II. SERIA PRACTICĂ

— Tematici teoretico-aplicative
— Situaţii tipice în proiectare, tehnologie
— Material tabelar şi grafic
— Îndrumar al activităţii, după criterii metodice şi eficiente

III. SERIA INIŢIERE

— Informare-instruire în domenii noi ce depăşesc pregătirea clasică
— Tratare sugestivă, cu aparat matematic accesibil
— Introduceri adresate specialiştilor
— Un ciclu separat de ABC-uri pentru cadre medii sau nespecialişti

IV. SERIA CONTINUĂ AUTOMATICĂ-MANAGEMENT CALCULATOARE (AMC)

— Număr mare de autori şi colaboratori
— Reflectarea evenimentelor vieţii tehnico-ştiinţifice, congrese, manifestări internaţionale etc.
— Cicluri de instruire
— Articole de sinteză, originale şi traduse, teme cu dezvoltare explozivă
— Articole cu bibliografii ample, indexabile multiplu
— Auxiliar preţios în pregătirea de cultură tehnică modernă

V. COLECŢIA AUTOMATICĂ-INFORMATICĂ

— Monografii succinte
— Documentare adîncită, variată
— Teme conturate
— Instrumente de lucru

VI. SERIA ELECTRONICĂ APLICATĂ

— Profil similar cu colecţia anterioară

VII. COLECŢIA „RADIO şi TELEVIZIUNE

— cărţi cu volum mic, în tiraj de masă, pentru radioamatori constructori, automatişti şi ciberneticieni

INFORMAŢII PENTRU TITLURILE CE APAR ÎN 1982-83, LA REDACŢIILE DE SPECIALITATE, EDITURA TEHNICĂ, PIAŢA SCÎNTEII 1, BUCUREŞTI.

Telefon 176010 interior 2100

În seria

AUTOMATICA MANAGEMENT CALCULATOARE (AMC)

vor apare succesiv în trimestrele II—IV 1982, numerele speciale 35, 36, 37, 38, 39, 40. Spre deosebire de volumele anterioare, aceste numere vor fi consacrate, chiar din titluri de pe coperte, unor tematici de actualitate ca

— Metodologia şi tehnicile de proiectare a sistemelor informatice

— Echipamentele de calcul fabricate în ţară

— Service, depanare, pentru echipamente de calcul şi de automatizare utilizate în ţară

— Congresul mondial IFAC, Japonia, 1981

— Cibernetica sistemelor industriale. Aplicaţii în robotică, automatizări suple, inteligenţă artificială

— La Editura tehnică, Piaţa Scînteii 1, Bucureşti, colectăm comenzi de la cititori şi întreprinderi pentru aceste volume. Comanda dvs. care trebuie să cuprindă acceptul dvs. de a plăti la livrarea volumelor ~ 20 lei/volum (120 lei pentru cele 6 volume ce apar pînă la sfîrşitul anului) trebuie să ne parvină pînă la 30 iunie 1982. Este indicat ca întreprinderile şi cititorii ce au avut comenzi în anii anteriori să le reînnoiască. Livrarea se va face la adresa indicată în comandă (fără ca banii să fie depuşi anterior), de către o unitate de desfacere, ca de ex. „Cartea prin poştă" Bucureşti.

- Se adresează acelor cititori – elevi, studenţi ş.a. – , care doresc să înveţe singuri programarea calculatoarelor, în limbajul COBOL.
- Fiecare din cele 22 lecţii este alcătuită din succesiuni de explicaţii, întrebări şi răspunsuri, care trebuie studiate activ, deci cu completarea răspunsurilor proprii, înainte de a consulta pe cele exacte, ale autorilor.
- Numeroase teste permit autoverificarea însuşirii cunoştinţelor, iar rezolvarea lor incorectă obligă la restudierea unor paragrafe indicate.

- Este prima carte românească în care se aplică metoda instruirii programate asupra programării structurate în COBOL.
- Aparţine ciclului de „instruiri programate" publicat în cadrul seriei INIŢIERE (Automatică • Informatică • Electronică • Management)

Lei 22

